Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/EP05/050599

International filing date: 10 February 2005 (10.02.2005)

Document type: Certified copy of priority document

Document details: Country/Office: EP
Number: PCT/EP2004/050111
Filing date: 10 February 2004 (10.02.2004)

Date of receipt at the International Bureau: 12 May 2005 (12.05.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

Europäisches Patentamt

European Patent Office

Office européen des brevets

PCT/EP2005/050599

EPO - DG 1

02. 05. 2005

(109)

Bescheinigung

Certificate

Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten internationalen Patentanmeldung überein.

The attached documents are exact copies of the international patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet international spécifiée à la page suivante.

Den Haag, den
The Hague,
La Haye, le 19 APR 2005

Der Präsident des Europäischen Patentamts
Im Auftrag
For the President of the European Patent Office
Le Président de l'Office européen des brevets
p.o.

EUROPÄISCHES PATENTAMT - EUROPEAN PATENT OFFICE - OFFICE EUROPEEN DES BREVETS
DEN HAAG - THE HAGUE - LA HAYE
237

C. v.d. Aa-Jansen

Patentanmeldung Nr.
Patent application no.
Demande de brevet n° PCT/EP 04/050111

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.:
Demande n°:

PCT/EP 04/050111

Anmelder:
Applicant(s):
Demandeur(s):

1. GENERALI INFORMATION TECHNOLOGIES SA - Manno, Switzerland
2. LENZARINI, Davide - Manno, Switzerland (US only)

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:

METHOD AND SYSTEM FOR SEAMLESS HANDOVER OF MOBILE DEVICES IN HETEROGENEOUS NETWORKS

Anmeldetag:
Date of filing:
Date de dépôt:

10 February 2004 (10.02.2004)

In Anspruch genommene Priorität(en)
Priority(ies) claimed
Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Bemerkungen:
Remarks:
Remarques:

The name of the applicant has been changed from the above-mentioned applicant to:
FORWARD INFORMATION TECHNOLOGIES SA - Manno, Switzerland

The registration of the change has taken effect on 20.09.2004

Method and System for Seamless Handover of Mobile Devices in Heterogeneous Networks

This invention relates to communication network access technologies and, more particularly, to a method and system for providing a seamless handover between heterogeneous networks, i.e. a transparent and automatic/semi-automatic switching between different network access technologies without interrupting active network applications or sessions.

Background of the Invention

In the last few years, the number of Internet users and the information offered has increased exponentially, together with the number of critical business and private activities relying on the network availability and the reliability of the connection. Even more people are used to access Internet frequently to make transactions (e.g. to buy goods and services, to book flights, to make banking transactions or trading), to access remote information (e.g. to read e-mails, to download files), to communicate (e.g. to chat, to make audio-video communications). That number is bound to continue to increase in the near future, since the growing range of IP (Internet Protocol) -capable mobile devices (e.g. PDAs (Personal Digital Assistant), smart-phones and laptops) and the growing availability of broadband wireless infrastructure (e.g. Wi-Fi [WIreless FIdelity for 802.11 network], GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunications System), EDGE (Enhanced Data-Rates for GSM Evolution),1XRTT (Radio Transmission Technology), CDMA2000 (Code Division Multiple Access), Bluetooth, etc.) are beginning to change our concept of Internet access from “static” or “nomadic” to “mobile”: nowadays the possibility of connecting to the Internet is no longer limited to a few network access points (e.g. at home, office, school or university) since it is possible to have network availability wherever the user is.

Ubiquitous access availability is important but the underlying concept is to provide reliable and continuous Internet access during the mobility of people nowadays, i.e. a seamless handover between different network access technologies. This is important since not all network technologies are suited to

cover similar need as range, access speed, etc. Therefore, network technologies as e.g. GPRS based on GSM (Global System for Mobile Communications), UMTS, WLAN (Wireless Local Area Network) or Bluetooth differ greatly in their characteristics and availability. However, co-existence of different network access technologies under the Internet Protocol (IP) brings problems when the user switches between access technologies and/or access providers: since IP addresses are assigned to a fixed location in the network, when we refer to mobile devices a new IP network address must be assigned with each change of network location (access technology and/or provider). This makes impossible a transparent, mobile access, leading to the IP applications to be restarted (therefore losing the current session), data packets to be lost and transmission rate to be slowed down.

Think about a busy manager who is working with his laptop in his office, connected via Ethernet or Wi-Fi to the company network. Suppose that he has a meeting in another city and he must leave the office to reach the local airport by taxi. Suppose he is using a Client/Server application requiring an always-on connection to complete a business-critical transaction. This manager has a problem. When he leaves the office to catch the taxi, the Wi-Fi/Ethernet connection will be lost and the application and the session that he is using will crash with possible loss of sensitive information. In order to complete his work, he will have to establish a GPRS/EDGE/UMTS/etc connection and to reload the Client application, repeating all the necessary authentication steps. At this point, he is able to continue his work, but only with the limited bandwidth offered by GPRS, EDGE or UMTS. When he reaches the airport, where a public hot spot is available, he could work with the larger and cheaper bandwidth of Wi-Fi; but in order to use the Wi-Fi technology, he has to switch from his current network connection to the Wi-Fi connection. This obviously involves all the above-mentioned problems, including the application crash and the new authentication. It is clear that a system should be capable of managing an automatic and transparent handover between different network access technologies and/or access providers without interrupting active network applications or sessions. This need is well known in the IT (Information Technologies) and Telecommunication worlds, which is also shown in the prior art by several patent publications e.g. by Nokia (EP 0 998 094 A2), Nortel (EP 1

089 495 A2), Swisscom (WO 02/103978 A2), etc. Another important need is that such a system should be completely flexible in order to be easily and quickly adapted to the variation of wireless network standards (e.g. in the Wi-Fi environment, the introduction of IEEE 802.11g or IEEE 802.11i (IEEE: Institute of Electrical and Electronics Engineers)), in order to be easily and quickly adapted to new network access technologies (e.g. based on the LEO [Low Earth Orbit] satellites), in order to be easily adapted to different OS (Operating System) (Windows, Linux, Symbian, PalmOS etc..) and to the future releases of such OS, and in order to be easily adapted to various mobile devices with different memories, computational capability and so on. Currently there are no standards providing a roaming service between the various kinds of wired/wireless networks. This lack of standards makes wired/wireless roaming a big issue if this problem is tackled at the lowest levels of the OSI-7 Layers Protocol Stack (Open System Interconnection). The prior art mentioned above satisfies the need and manages the seamless handover proposing a mechanism operating at the lowest OSI layers (L2 and L3). As said before, this can represent a relevant issue because of the vast amount of low-level work that must be done on the OS, with a negative impact on the flexibility and portability of the solution.

Virtually all networks in use today are based in some fashion on the Open Systems Interconnection (OSI) reference model of the ISO (International Organization for Standardization) standards. The core of this standard is the OSI Reference Model, a set of seven layers that define the different stages that data must go through to travel from one device to another over a network. Referring to Figure 1, an overall scheme for the standard OSI-7 Layers Protocol Stack is shown. The OSI standard is the only internationally accepted framework of standards for communication between different systems made by different vendors. The OSI layers are: Physical Layer (L1), which corresponds to the physical network interface, deals with the physical means of transmitting data over communication lines (referring in a network environment to various Network Interface Cards). On L1 each node of a network, for example, with a packet-switched interface has an unambiguous network address, these network addresses being called a Data Link Control (DLC) address or a Media Access Control (MAC) address. In the case of networks which conform to the IEEE 802

standard (such as Ethernet, for example), the DLC addresses are usually called MAC addresses. To be called a DLC address, an address must fulfill at least the OSI reference model. In other words, a DLC address, or respectively a MAC address, is a hardware address that identifies the node or respectively the physical network interface unambiguously in the network. Some protocols, such as Ethernet or Token Ring, for example, use the DLC/MAC address exclusively, i.e. they cannot communicate with the respective node without this address. A circuit-switched interface, on the other hand, has no such DLC or MAC address, i.e. thus also no corresponding identification DLCI (DLC Identifier). Examples of protocols using circuit-switched interfaces are inter alia PPP (Point to Point Protocol), SLIP (Serial Line Internet Protocol) or GPRS (Generalized Packet Radio Service). Data link Layer (L2), which is concerned with procedures and protocols for operating the communication lines. Network Layer (L3), which provides switching and routing technologies, creating logical paths for transmitting data from node to node. This information may include network or Internet protocol addresses for communication nodes. Transport Layer (L4), which defines the rules for information exchange, e.g. information about various network protocols. Session Layer (L5), which establishes, maintains and ends communication with the receiving device. Presentation Layer (L6), which takes the data provided by L7 and converts it into a standard format that the other layers can understand. Application Layer (L7), which supports application and end-user processes. This is the layer that actually interacts with the operating system or application whenever the user chooses to transfer files, read messages or perform other network-related activities.

As mentioned, in the prior art we can find the following three patent applications, which can be regarded as representing the prior art for the issue of avoiding the client application shutdown during the wireless network connection switches. These are WO 02/103978 A2 (Swisscom Mobile AG), EP 1 089 495 A2 (Nortel Networks Limited) and EP 0 998 094 A2 (Nokia Mobile Phones LTD). All these patent applications make use of the concept of Mobile IP as described in IP Mobility Support – IETF RFC 2002 (C. Perkins - IBM IP Mobility Support - IETF RFC 2002 - October 1996). Internet makes use of the IP (Internet Protocol) to route data packets (datagrams) from the source to the destination. The source and the destination must have an IP address unique in Internet in

order to be reached, something like the telephone number in the telephony world. When the destination address of the data packets is a mobile node this means that a new IP network address must be assigned with each change of network location, which makes transparent mobile accesses impossible. These mobility problems were solved by the Mobile IP standard of the IETF. Mobile IP allows the mobile node to use two IP addresses. One of these addresses is the normal, static IP address (home address), which indicates the location of the home network, whereas the second is a dynamic IP care-of address, which provides information about its current point of attachment to the Internet. The assignment of the two addresses allows the IP data packets to be rerouted to the correct, momentary address of the mobile node. The Mobile IP provides for registering the care-of address with a Home Agent. The Home Agent is normally a fixed network node, which administers the two addresses of the mobile node (home address and care-of address) and reroutes or routes the corresponding data packets: it sends datagrams destined for the mobile node through a IP tunnel to the care-of address. After arriving at the end of the tunnel, each datagram is then delivered to the mobile node.

Unfortunately, the Mobile IP of the IETF does not solve all the mobility problems: if, for instance, a user would like to switch between two different network interfaces while an IP application is running, the IP connection is interrupted at the moment when he leaves the old network link. This connection is interrupted at least until the new location, i.e. the new care-of address, is known and it has been registered at the so-called Home Agent. If the interruption time for the change exceeds the time-out delays specified e.g. in the TCP (Transfer Control Protocol), the IP connection is of course interrupted anyway. Even when the interruption time lies within the time-out delays specified e.g. in the TCP, however, the IP applications are not able to maintain the connection if a physical network interface is not permanently available. Thereby IP applications normally have to be restarted after a network connection switch in order to access a new IP data tunnel. WO 02/103978 A2 provides a method and a system to avoid the interruption of service in case of network connection switch with a mechanism operating at layer 3 (Network layer) of the OSI-7 Layers Protocol Stack. In Figure 2 the reference numeral 10 refers to the mobile device, 11 is the application layer of the IP applications and

12 refers to the TCP layer. The solution of WO 02/103978 A2 is based on the three main layers or respectively main modules 131 to 134 which are designated jointly as mobile module by the reference numeral 13. The first layer consists of a mobile IP module 131 and/or an IPSec module 132. The second layer is the virtual network interface 133 of the solution and the third layer is an interface administration module 134 to handle the physical network interfaces 14 - 17. Finally, reference numerals 21 to 24 accordingly stand for the various heterogeneous networks and 30 designates the usual, worldwide IP backbone network. The Virtual IP Network Interface (133 in Figure 2) and the Interface Administration Module (134 in Figure 2) make transparent to the Client applications (11-12 in Figure 2) the care-of IP address changing. The main drawback of WO 02/103978 A2 is that, operating at layer 3, the implementation requires a great deal of low-level work for each supported operating system. This vast amount of work reduces the flexibility of this solution in case of variation in wireless networks standards or in case of introduction of new wireless networks. WO 02/103978 A2 is based on the concepts of Mobile IP, and it solves its problem of IP applications restart in case of a network connection switch. Consequently all the coordination issues between the mobile node and the home agent have to be considered and implemented by this patent. Furthermore WO 02/103978 A2 requires that the Virtual IP Network Interface be under a custom Mobile IP and/or IPSec module (131-132 in Figure 2) that provides Mobile IP and security features (authenticity of the interlocutors, confidentiality of the data exchanged and hashing systems to check whether the data exchanged have been modified during the transport by an unauthorized third party). Using a custom IPSec module all the above-mentioned security features have to be implemented, thus the widespread security commercial products operating at transport (L4) or network (L3) or lower level (L2, L1) can't be used. This patent does not take care of how the network connection is made and, operating at layer 3, an automatic or semi-automatic/assisted way to make the connection is difficult to be achieved. The solution of WO 02/103978 A2 is limited in its architectural features by Mobile IP concepts, on which it's based.

The mentioned document EP 1 089 495 A2 of Nortel shows a method and a system to make a change of the physical interfaces without the active IP applications being interrupted or having to be restarted because their

link to the original interface has been lost (see Figure 3). As Figure 3 shows, the solution is based on a typical OSI-7 Layer Protocol Stack where reference numeral 16 designates the Network Layer (L3). Reference numeral 60 stands for an NAA (Network Access Arbitrator), 62 are the network adapters (NICs), 64 are the adapter drivers and 36 is a specific computer hardware platform. Nortel proposes an NAA (Network Access Arbitrator) 60 to reroute, via a single fixed MAC (Media Access Control) address of the so-called primary NIC (Network Interface Card) 62, the various MAC addresses of the individual configurable physical network interfaces available. The NAA 60 connects the layer 2 (Data Link layer) of the available NICs 62, and it reroutes the data packets from the primary NIC 62 to the corresponding MAC address of a further network interface (secondary NIC) 62. The NAA 60 provides a virtual adapter driver, and it requires that at least one physical interface with a MAC address must be permanently available. The major drawback of the Nortel invention is that it is sensitive to the definition of the network interface hardware-related address. If the address does not correspond to the IEEE 802 standard (MAC addresses) and if the new address standard has not been explicitly defined beforehand in the NAA, the NAA does not function with these interfaces since it can no longer reroute the MAC addresses. Another disadvantage arises from the explicit use of the MAC addresses: circuit-switched interfaces (GPRS, PPP (Point-to-Point Protocol), SLIP (Serial Line Internet Protocol)) do not have any corresponding MAC or network addresses. Since the NAA is able to register only devices with MAC addresses in order to reroute the data packets, circuit-switched interfaces are not available to the NAA even though their connection to the IP layer should also be possible. A further disadvantage of EP 1 089 495 A2 has its origin in being based on the concepts of Mobile IP. In fact, EP 1 089 495 A2 solves the problem of IP applications restart in case of network connection switch, but consequently, all the coordination issues between the mobile node and the home agent have to be considered and implemented by this solution.
Additionally, this solution does not take care of how the network connection is made which can be problematic for many applications. Like the solution of WO 02/103978 A2, the solution of EP 1 089 495 A2 is limited in its architectural features by Mobile IP concepts, on which it's based.

Finally EP 0 998 094 A2 of Nokia provides another method and system to avoid the interruption of service in case of network connection switch. The mechanism of the solution operates between layer 2 (Data link layer) and layer 3 (Network layer) (see Figure 4). In Figure 4 PD designates a Protocol Driver, NT refers to the Windows NT standards of Microsoft and NISD is a Network Interface Selection Driver. The main drawback of this solution is that, operating between layer 2 and 3, the implementation requires a great deal of low-level work for each supported operating system. This vast amount of work reduces the flexibility of this solution in case of variation in wireless networks standards or in case of introduction of new wireless networks. Again, EP 0 998 094 A2 is based on the concepts of Mobile IP to solve the problem of IP applications restart in case of network connection switch. Consequently all the coordination issues between the mobile node and the home agent have to be considered and implemented by this solution. Like the solution of WO 02/103978 A2 and EP 1 089 495 A2, additionally, the solution of EP 0 998 094 A2 is limited in its architectural features by Mobile IP concepts, on which it's based.

Summary of the Invention

It is an object of this invention to propose a new method and system for seamless handover of mobile devices in heterogeneous networks. In particular the switching from one network connection to another should be carried out without interruption of the IP applications and makes possible an uninterrupted continuation of the program course also with real-time applications, if applicable, without being dependent upon specific protocols or network technologies or operating systems. Therefore, it is an object of this invention to provide a method and a system capable of managing, without being dependent upon different protocols or network technologies or operating systems, an automatic/semi-automatic and transparent handover between different network access technologies and/or access providers without interrupting active network applications or sessions.

This object is attained according to the present invention through the elements of the independent claims. Further preferred embodiments follow, moreover, from the dependent claims and from the description.

In particular, this object is achieved through the invention in that for seamless handover of mobile devices in heterogeneous networks in which method a mobile device (as e.g. a mobile device in a IP network) is moved between different topological network locations and/or transmits and/or receives data by means of different network access technologies without the data transfer of active Client IP applications being interrupted, in that a Client IP application of the mobile device makes an IP request to a client-service module on the mobile device, in that the client-service module creates a logical data tunnel with a server-service module and reroutes the Client IP request to it, and in that the server-service module reroutes the Client IP request to a Server IP application that creates an IP socket for the Client IP request to handle the exchange of IP data packets with the Client IP application. The client-service module can e.g. provide at least a server application emulation interface comprising IP sockets to be used by the Client IP application and a client application emulation interface comprising IP sockets to be used by the server-service module. The server-service module can e.g. provide at least a server application emulation interface comprising IP sockets to be used by the client-service module and a client application emulation interface comprising IP sockets to be used by the Server IP application. The server application emulation interface of the client-service module can e.g. comprise IP sockets bound to the loopback IP address and the client application emulation interface of the client-service module can e.g. comprise IP sockets bound to the mobile device IP address used by the client-service module. The Client IP application can e.g. create for the Client IP request a client application request socket connected to the client-service module, the client-service module can provide a server service emulator server socket accepting the Client IP request, the client-service module can create a client request emulation socket connected to the server-service module to reroute the Client IP request, the server-service module can provide a server service emulator server socket accepting the rerouted Client IP request of the client-service module, the server-service module can create a client request emulation socket connected to the Server IP application to reroute the Client IP request, and the Server IP application can provide a server service server socket accepting the Client IP request of the client application. The server-service module can e.g. run on a network device different to the network device on which runs the Server IP application. The

client-service module can e.g. run on a first network device and the Client IP application on a second different network device, whereas the first network device and the second network device are connected via a broadband network connection forming a logical mobile unit together. In particular, the network connection can e.g. consist of a broadband network connection comprising Bluetooth and/or Ethernet and/or Wi-Fi. The Server IP application can e.g. send its IP response to the server-service module, wherein the Server application IP response path is the mirror image of the Client IP request path.

Further if a Server IP application makes an IP request to the server-service module the Server IP application can e.g. create for the Server IP request a server application request socket connected to the server-service module, the server-service module can provide a client service emulator server socket accepting the Server IP request, the server-service module can create a server request emulation socket connected to the client-service module to reroute the Server IP request, the client-service module can provide a client service emulator server socket accepting the rerouted Server IP request of the server-service module, the client-service module can create a server request emulation socket connected to the Client IP application to reroute the Server IP request, and the Client IP application can provide a client service server socket accepting the Server IP request of the server application. The client-service module and/or the server-service module can be e.g. pure Layer 7 application according to the Open System Interconnection Structure of the International Organization for Standardization. The client-service module and/or the server-service module can e.g. be created as cross platform modules. The client-service module and/or the server-service module can e.g. be composed at least in parts by Java modules. A plurality of client-service modules of one or more mobile devices can e.g. be assigned to the same server-service module and the client application emulation interface sockets of the server-service module can be bound to different Virtual IP addresses created and/or allocated by it. A plurality of Server IP applications resident on one or more network devices can e.g. be assigned to the same server-service module. A plurality of Client IP applications resident on one or more network devices can be assigned to the same client-service module. The client-service module can e.g. be assigned simultaneously to a plurality of server-service modules resident on one or more

network devices. For different services the client-service module can e.g. be assigned to the plurality of server-service modules, whereas each service is handled by a corresponding server-service module.

Additionally the client-service module can e.g. check the mobile device for available physical network interfaces and creates a lookup table with the available and configurable physical network interfaces, the client-service module creating the data tunnel via one of the physical network interfaces contained in that lookup table. The client-service module of the mobile device can e.g. check periodically for available physical network interfaces. With a change or an update of the physical network interface of the mobile device used by the client-service module, the connection of the data tunnel to the network can e.g. be kept up and updated by means of the client-service module using the lookup table. The client-service module automatically changes and updates the physical network interfaces of the mobile device on the basis of information from the lookup table. The client-service module can e.g. switch from an old IP address to a new IP address by means of a switching procedure which keeps up and updates the data tunnel exchanging IP control packets with the server-service module in order to provide a seamless handover. The client-service module can e.g. switches suddenly from an old IP address no more available to a new IP address by means of a switching procedure which restores and updates the data tunnel exchanging IP control packets with the server-service module in order to provide a seamless handover. The criteria for the automatic change and/or update of the physical network interface by the client-service module can e.g. be defined by the user. A change or an update of the physical network interface used by the client-service module can be initiated by the user. The available physical network interfaces can also e.g. be configured dynamically or statically. If the network connection used by the client-service module is interrupted, IP data packets coming from the Client IP applications can e.g. be buffered in a data buffer of the client-service module until the data tunnel to the server-service module is restored. If the network connection to the client-service module is interrupted, IP data packets coming from the Server IP applications can e.g. be buffered in a data buffer of the server-service module until the data tunnel to the client-service module is restored.

In particular, this object is achieved through the invention in that a mobile device is moved between different topological network locations and/or transmits and/or receives data by means of different network access technologies without the data transfer of the active IP applications being interrupted, in that a Client IP application of the mobile device, needing some server services, makes an IP request to a client-service module (called CNAPT: Client Network Address and Port Translator) on the mobile device, believing to interact directly with the proper Server IP application, in that the client-service module creates a logical data tunnel with the proper server-service module (called SNAPT: Server Network Address and Port Translator) and reroutes the Client IP request to it, in that the server-service module forwards the client IP request to the proper Server IP application, in that the Server IP application, believing to interact directly with the Client IP application, sends its IP response to the local server-service module, in that the Server application IP response path is the mirror image of the Client IP request path and, finally, in that an eventual IP request made by the Server application, needing some client services, is handled in a mirror-like manner as the Client application IP request. In particular, the movement between different topological network locations and/or transmission and/or receiving data by means of different network access technologies can, for instance, comprise a change of the physical network interface and/or a change among different networks technologies, such as e.g. Ethernet, Bluetooth, mobile radio networks (GSM, EDGE, GPRS, CDMA200, UMTS, etc.) or WLAN (Wireless Local Area Network), or also a topological location change within the same network technology, for example a device linked to an Ethernet network that migrates to another Ethernet network or a device linked to a Wi-Fi HotSpot that migrates to another Wi-Fi HotSpot. To be understood as a mobile device are inter alias all possible devices (laptops, PDAs, smart-phones etc) equipped with one or more different physical network interfaces, which can support a plurality of different network standards. The physical network interfaces of the mobile device can comprise e.g. interfaces for Ethernet or for another wired LAN, Bluetooth, GSM, GPRS, EDGE, CDMA2000, UMTS and/or WLAN, etc. The interfaces can be not only packet-switched interfaces, as used directly by network protocols such as e.g. Ethernet or Token Ring, but also circuit-switched interfaces which can be used by means of protocols such as e.g. PPP (Point-to-Point Protocol), SLIP (Serial Line Internet

Protocol) or GPRS (Generalized Packet Radio Service), i.e. which interfaces do not have, for example, any network addresses such as a MAC or a DLC address.

It should be stated here that, besides the method according to the invention, the present invention also relates to a system and a computer program product for carrying out the method.

Various embodiments of the present invention will be described in the following with reference to examples. The examples of the embodiments are illustrated by the following attached figures:

Figure 1 shows a block diagram illustrating schematically the OSI-7 Layers Protocol Stack as defined in the prior art.

Figure 2 shows a block diagram illustrating schematically the architecture of WO 02/103978 A2 of the prior art.

Figure 3 shows a block diagram illustrating schematically the architecture of EP 1089495 A2 of the prior art.

Figure 4 shows a block diagram illustrating schematically the architecture of EP 0998094 A2 of the prior art.

Figure 5 shows a block diagram illustrating schematically a Client/Server application using Internet to exchange data.

Figure 6 shows a block diagram illustrating schematically the Client application using a local network connection and the Internet to reach its Server application.

Figure 7 shows a block diagram illustrating schematically the Client application sending a request of service A to its appropriate Server application.

Figure 8 shows a block diagram illustrating schematically the Client NAPT (CNAPT) module according to the invention.

Figure 9 shows a block diagram illustrating schematically the Client NAPT (CNAPT) module and the Client application installed on different devices directly linked to form a PAN (Personal Area Network).

Figure 10 shows a block diagram illustrating schematically the Server NAPT (SNAPT) module according to the invention.

Figure 11 shows a block diagram illustrating schematically an embodiment according to the invention, the "Basic configuration".

Figure 12 shows a block diagram illustrating schematically an embodiment variant according to the invention, the "Multi-Site configuration".

Figure 13 shows a block diagram illustrating schematically an embodiment variant according to the invention, the "Client PAN configuration".

Figure 14 shows a block diagram illustrating schematically an embodiment variant according to the invention, the "Multi-Server LAN configuration".

Figure 15 shows a block diagram illustrating schematically an embodiment variant according to the invention, the "Multi-Server Internet configuration".

Figure 16 shows a block diagram illustrating schematically an embodiment variant according to the invention, the "Client PAN Multi-Server Internet configuration".

Figure 17 shows a block diagram illustrating schematically an embodiment according to the invention based on the "Basic configuration" where the Client application uses a local network connection and the Internet to reach its server application.

Figure 18 shows a block diagram illustrating schematically the Client NAPT of an embodiment according to the invention based on the "Basic configuration".

Figure 19 shows a flowchart illustrating schematically the method used by "Search Activity" to search for alternative network providers.

Figure 20 shows a flowchart illustrating schematically the method used by CNAPT module to retrieve an Internet connection at its start.

Figure 21 shows a block diagram illustrating schematically the Server NAPT of an embodiment according to the invention based on the "Basic configuration".

Figure 22 shows a block diagram illustrating schematically a Client/Server interaction by a system according to the invention.

Figure 23 shows a block diagram illustrating schematically the CNAPT module stopping forwarding service requests and sending a SWITCH_STEP1 packet.

Figure 24 shows a block diagram illustrating schematically the SNAPT module stopping forwarding data packets and sending an acknowledgement packet and the LAST_MESSAGE_BEFORE_REDIRECTION packets.

Figure 25 shows a block diagram illustrating schematically the CNAPT module sending the LAST_MESSAGE_BEFORE_REDIRECTION packets.

Figure 26 shows a block diagram illustrating schematically the SNAPT module sending the OK_TO_REDIRECTION packet.

Figure 27 shows a block diagram illustrating schematically the CNAPT module sending the CLIENT_SERVICE_READY_FOR_REDIRECTION packet.

Figure 28 shows a block diagram illustrating schematically the SNAPT module sending the SERVER_SERVICE_READY_FOR_REDIRECTION packet.

Figure 29 shows a block diagram illustrating schematically the CNAPT module sending the SWITCH_STEP2 packet, destroying the Client request emulation sockets and replacing the control socket.

Figure 30 shows a block diagram illustrating schematically the CNAPT module recreating the Client request emulation sockets and binding them to the new IP address.

Figure 31 shows a block diagram illustrating schematically the SNAPT module sending the ALL_REDIRECTED packet.

Figure 32 shows a block diagram illustrating schematically the IP transition phase completed without any interruption of service.

Figure 5 to 11 illustrate an architecture that can be used to achieve the invention. The invention can be used by any Client application that is connected through an IP network (e.g. Internet or any wired/wireless Intranet) to its Server application (Figure 5). The method and system according to the invention comprise one or more Client applications 11, running on a mobile device 10, that are connected via a local network provider 30/31/32/33 and through an IP network 34, e.g. the worldwide backbone network called Internet or any wired/wireless Intranet, to a Server application 21 (Figure 6). The networks implemented by providers 30/31/32/33 can comprise Ethernet or another wired LAN (Local Area Network), Bluetooth, GSM (Global System for Mobile Communication), GPRS (Generalized Packet Radio Service), EDGE (Enhanced Data-Rates for GSM Evolution), 1XRTT (Radio Transmission Technology), CDMA2000 (Code Division Multiple Access), UMTS (Universal

Mobile Telecommunications System) and/or WLAN (Wireless Local Area Network), etc. To be understood as mobile device 10 are, inter alias, all possible devices (laptops, PDAs, smart-phones etc) equipped with one or more different physical network interfaces 40 to 43, which can also support the different network standards of the various network providers 30 to 33. The physical network interfaces 40 to 43 of the mobile device 10 can comprise e.g. interfaces for the mentioned networks as e.g. Ethernet or another wired LAN, Bluetooth, GSM, GPRS, EDGE, CDMA2000, UMTS and/or WLAN, etc. The invention provides any Client application with the best local network connection to the IP network 34 in term of bandwidth, reliability and cost effectiveness among all the wired/wireless network access technologies and/or access providers available at a certain time and location 30/31/32/33, managing the switch (when needed or convenient) in a transparent automatic and/or semi-automatic way without interrupting active network applications or sessions. In particular, the switch between the networks can comprise a change of a physical network interface among different networks technologies, such as e.g. Ethernet, Bluetooth, mobile radio networks (GSM, GPRS, EDGE, CDMA2000, UMTS, etc.) or WLAN, or also a topological location change within the same network technology, for example a device linked to an Ethernet network that migrate to another Ethernet network or a device linked to a Wi-Fi HotSpot that migrate to another Wi-Fi HotSpot. In particular the invention is suited for client network applications running on a mobile device 10 which is able to hold simultaneously at least two different kind of network connection (GSM, GPRS, EDGE, CDMA2000, UMTS, Satellite Links, Wi-Fi, LAN, PSTN [Public Switched Telephone Network], xDSL as ADSL [asymmetric digital subscriber line] or SDSL [symmetric digital subscriber line] etc): for instance mobile devices with a GPRS modem and an independent Wi-Fi adapter or mobile devices with a Wi-Fi adapter and an independent UMTS modem or so on. However, it is important to note that the invention also works with mobile devices able to hold only one network connection at a time (e.g. a laptop that possesses only one slot for insertion of a PCMCIA network card (PCMCIA: Personal Computer Memory Card International Association)). According to the invention the client application and its server exchange data via an IP network, such as e.g. the Internet, using exactly one network connection at a time.

Suppose that the mobile device using the client application is able to provide N different network connections through N network adapters (with N≥1) (40-43). The server application may provide one or more services and for each of them a server socket 29 listens for client application requests (Figure 6). Suppose further that the mobile device is e.g. connected to Internet via the network provider X [31] and has the IP address "ClientIP_X". Besides, suppose that a second network provider, provider Y [32], is available at the same time. The client application now uses a socket to send its server a service request. The server replies with packets having "ClientIP_X" as destination IP address (Figure 7). If Provider X becomes unavailable (slowly, e.g. the Provider X is a Wireless provider and the mobile device is slowly leaving its coverage area, or suddenly, e.g. the LAN/Ethernet cable is suddenly unplugged), while the other Provider Y remains still available, a new network connection through the Provider Y must be established, in order to maintain active the client application and the current session. When established, the old network connection, if already up, should be closed because it will no longer be available. The problem is that with the new network connection, the mobile device will be assigned a new IP address (therefore changing its IP address from "ClientIP_X" to "ClientIP_Y"): due to such a change, the service response can no longer reach the client application, resulting in the application/session crash and interruption of the service.

The invention avoids any interruption of service without any modification of the providers' infrastructure. One of the purposes of invention is to make the IP address change of the mobile device totally transparent to the client and server applications. The invention acts as a middleware, making the client application to believe that it is running either on the same machine as the server application or in a machine directly connected to the server (depending on the configuration adopted). This is achieved through the client-service module 12, further called client NAPT (CNAPT: Client Network Address and Port Translator module) (Figure 8), and a server-service module 22, further called Server NAPT (SNAPT: Server Network Address and Port Translator Module) (Figure 10). By means of the CNAPT module and SNAPT module, if the provider X becomes unavailable (slowly or suddenly) while the network provider Y remains still available, and if there is enough time for the IP transition

phase (that is to say the client and the server applications do not go into timeout), the switch between "ClientIP_X" and "ClientIP_Y" is totally transparent to the client and server applications that continue their execution without any session crash and/or interruption of service. Note that the IP transition phase can be generated also by a temporary interruption of the Internet connection provided by the network provider X, that however remains still available; in fact that temporary interruption may cause a modification on the assigned IP address, e.g. from "ClientIP_X" to "ClientIP_X1". During the IP transition phase: (1) The CNAPT stops forwarding all the outgoing IP packets generated by the Client application. They are buffered by the CNAPT and they will be forwarded at the end of the IP transition phase. (2) The SNAPT stops forwarding all the outgoing IP packets generated by the Server application and directed to the switching CNAPT. They are buffered by the SNAPT and they will be forwarded to the switching CNAPT at the end of the IP transition phase.

Comprising the CNAPT module and the SNAPT module the invention is able to act as a layer 7 relay system with a high level of flexibility. The CNAPT module processes the client requests and relays them to the SNAPT module. The SNAPT module processes the client requests and, at its turn, relays them to the server application. The server response path is the mirror image of the client request. The CNAPT module and the SNAPT module comprise the necessary infrastructure, including hardware and software components and/or units, to achieve a described method and/or system according to the invention.

All operations of the CNAPT module and the SNAPT module are performed according to the invention at application level (layer L7 of the OSI-7 Layers Protocol Stack): operating at the highest level of abstraction, i.e. of layers, it has the advantage that all activities can be easily and quickly adapted to the variation of wireless networks standards, to new network access technologies, to the different OSs (Operating Systems) and to future releases of such OSs. Moreover, operating at layer 7, the invention has many further advantages compared to the methods and systems of prior art: (1) It does not have to take care of Mobile IP concepts and implementation; i.e. it works in a transparent way over Mobile IP (v4 or v6) exactly as it works over IPv4 or IPv6.

(2) It can use all the widespread commercial products at transport (L4) or network (L3) or lower level (L1, L2) to manage security and provide all the security features. Particularly, it does not require the implementation of any custom security feature: it can work in a transparent way over IPSec (IP Security by IETF (Internet Engineering Task Force)), L2TP (Layer Two Tunnelling Protocol) or other VPN (Virtual Private Network) protocols. (3) It does not care what kind of network connection is used, and the invention thereby works indifferently in circuit-switched and packet-switched networks. (4) It provides an automatic and/or a semi-automatic/assisted way to make network connections. The comparison of this invention with the prior art highlights its innovative idea and its advantages. Operating at layer 7, this invention provides an extreme flexible and simple way to solve the problem of the interruption of service in case of network connection switch.

According to the invention, the Server IP address is modifiable in the client application (i.e. the Server IP address is not hard coded). The SNAPT module comprises in an appropriate memory unit at least the server IP address (or, in turn, the servers' IP addresses in case of multiple server applications, running on different machines, managed by the same SNAPT module), the server services' ports and the server services' type (connection-oriented, like services based on TCP (Transmission Control Protocol), or connectionless, like services based on UDP (User Datagram Protocol)) of its managed server application. The SNAPT module comprises means to emulate server services 29 providing a set of emulation services 26 operating on emulation ports different from the real services port in order to avoid bind errors (Figure 10). Each CNAPT module comprises a unique ID (IDentification Number) that differentiates it from the other CNAPT modules. The CNAPT module further comprises in a memory unit at least the SNAPT IP address (or, in turn, the SNAPTs' IP addresses in case of a CNAPT module simultaneously connected to more than one SNAPT module), the server services' ports and type (connection-oriented or connectionless) of the server application managed by the SNAPT and the SNAPT module services' emulator ports. The CNAPT module also comprises in a memory unit the client IP address (or, in turn, the clients' IP addresses in case of multiple client applications, running on different machines, managed by the same CNAPT module), the eventual client services'

ports and the client services' type (connection-oriented or connectionless) of its managed client application. Finally the CNAPT module comprises means to emulate the eventual client services 111 providing a set of emulation services 17 operating on emulation ports different from the real services ports in order to avoid bind errors (Figure 8). The SNAPT module comprises, for each CNAPT module that provides client services, the client services' ports, the CNAPT services' emulator ports and the client services' type (connection-oriented or connectionless). These data are grouped by CNAPT module ID. If the client services are used, the SNAPT module must comprise also a set of virtual IP addresses 23 (Figure 10) belonging to the same network to which it belongs (i.e. if the SNAPT module has the IP 192.168.102.150 and belongs to the network 192.168.102/24, it must have a set of virtual IP addresses belonging to 192.168.102/24 like 192.168.102.151, 192.168.102.152 and so on). Those addresses are used to accept simultaneous connections of different CNAPT modules managing client applications providing their client services on the same port, in order to avoid bind errors on that port. This is the case e.g. of n identical CNAPT modules managing a client application providing a client service on the port X, connected to a single SNAPT module. A different virtual IP address is used for each of those identical CNAPT modules. Obviously the number of virtual IP addresses must be equal to the number of those identical CNAPT modules that can access the SNAPT module simultaneously. During normal activity the SNAPT module knows the CNAPT module ID and the current IP of all the connected CNAPT modules.

As explained before, the invention comprises the possibility that the client application could also provide a set of services (connectionless or connection-oriented), the client services, that can be used by the server application for "PUSH" or "Publish/Subscribe" paradigms. For instance, the client application contacts the server application to register for some information updates, so when new information becomes available the server application may "push" them to the subscribed client applications on their services' ports. From now on, to simplify the explanation and the drawings, only the services provided by the server application are considered: it is easy to understand that client services are managed by the invention in a mirror-like manner to the server services.

It is important to note that the invention does not require any modification of the client applications source code if: (a) The server IP address is not hard coded (that is to say the server IP address is not set directly in the source code). It must be possible to set the server IP address in a configuration file or at run time, with user 1 interaction (Figure 6), in some client application input mask. (b) The source IP address of the eventual client services provided is not hard coded. It must be possible to set the source IP address of these client services to “loopback” address (the special address “127.0.0.1” for IPv4 and the “::1” for IPv6) in a configuration file or at run time, with user 1 interaction, in some Client application input mask. This is necessary because if the source IP address is set automatically to the mobile device IP, when this IP address changes for a switch phase the client services would be interrupted. This constraint is not necessary if the client applications 11 are running on devices 10 different from the CNAPT module device 13 (Figure 9). In fact, in this case the source IP address is the client application device PAN IP address 18 (PAN: Personal Area Network) and it does not change during a switch phase (only the CNAPT module Internet IP address 19 changes during a switch phase). The invention does not require any modification of the Server application source code unless the client IP address, retrieved from the incoming packets, is used for critical internal activities (for instance in case of authentication based on the client IP address).

The Client NAPT (CNAPT) module according to the invention can be realized by means of appropriate software and/or hardware components. The CNAPT module comprises means to emulate server applications on the client side 14 and means to emulate client applications on the network side 15, e.g. Internet side (Figure 8). The CNAPT module creates for each server service, a server socket 16 on the client side. This server socket listens on the real server service port (server service emulator server socket). For each server service request, the CNAPT module provides a client request emulation socket 121 on the Internet side. This socket relays packets to the right service emulator server socket provided by SNAPT. For each client service 111, the CNAPT module provides a server socket 17 on the network side, e.g. on the Internet side. This server socket listens on (i.e. waits for a signal on) the client service emulator port (client service emulator server socket). This port is different from the client

service real port in order to avoid a bind error. For each client service request, the CNAPT module provides a server request emulation socket 123 on the client side. This socket relays packets to the real Client server socket.

The CNAPT module not only acts as a layer 7 relay system. It also launches a "decision task" module in order to provide the client application with the best network connection, e.g. Internet connection, in term of bandwidth, reliability and cost effectiveness. The "decision task" module can be realized as a software and/or hardware module and has two main activities: (1) It continuously verifies the current network connection reliability and performance (Check Activity). (2) It continuously searches for new network providers (Search Activity). At any time, the user 1 can ask to the "decision task" module to switch to another available network provider. This is useful when the current network connection is a wired connection and the user 1 wants to switch to a wireless connection unplugging for instance the Ethernet cable. Note that the "decision task" can launch the Search and the Check activities only if it is running on a mobile device that is able to hold simultaneously at least two different kind of network connection (GSM, GPRS, EDGE, UMTS, Satellite Links, Wi-Fi, LAN, PSTN, ADSL ... etc). If that mobile device is able to hold only one network connection at a time (e.g. a laptop that possesses only one slot for insertion of a PCMCIA network card), the Search activity can't be launched and the Check activity only verifies the reliability of the current Internet connection signalling to the user 1 its eventual unavailability. In this case, the switch from one network provider to another (or also, in case of temporary interruption, to the same network provider if it is became available in the meanwhile, e.g. there was only a temporary problem) can be decided only by the user 1 with a specific request to the "decision task".

The Check Activity as a part of the "decision task" module verifies the reliability and the performance of the current network connection, e.g. Internet connection, every Y ms. If the reliability/performance indexes go down some specified "warning thresholds" (that can be set by the user 1) or if the current Internet connection has been interrupted, the "Check Activity" searches for new network providers in a similar way as the Search Activity (or trying to retrieve a new Internet connection from the same network provider if it is became

available in the meanwhile, e.g. there was only a temporary problem). It can work in two ways: manual and automatic mode. In both the modes if it does not find any other network providers, it signals to the user 1 that the current network connection will be no longer available. In the manual mode, if it finds at least one alternative provider, it suggests the user 1 to switch to the better alternative. If and only if the user 1 does not authorize the switch and the current network connection indexes go down some specified “critical threshold” (that can be set by the user 1), the Check Activity, in order to avoid any interruption of service, switches to an alternative network provider available (preferring the provider allowing a totally automatic connection procedure) and it signals to the user 1 that the current network connection will be no longer available. In the automatic mode, if the Check Activity finds at least one alternative provider, it automatically decides on the network provider switch (preferring the alternative provider which allows a totally automatic connection procedure). It may perhaps require some user 1 interaction to establish a semi-automatic/assisted network connection. During the switch, the Check Activity avoids any interruption of service by the method described in this invention. After the switch has been made the old network connection, e.g. Internet connection, will be closed if it is still open.

The Search Activity as another part of the “decision task” module searches, without disturbing the current connection, every X ms for other available networks providers (Figure 19). It can work in two ways: manual and automatic mode. In the manual mode the user 1 has to choose the available network provider that he prefers. When the Search Activity finds at least one network provider that could provide a network connection, e.g. Internet connection, better than the current one, it asks the user 1 if he wants to switch. In the automatic mode, the Search Activity automatically decides on the network provider switch (the decision is made using a set of parameters that can be modified by the user 1). It may eventually require some user 1 interaction to establish a semi-automatic/assisted network connection. During the switch, the Search Activity avoids any interruption of service by the method described this invention. After the switch has been made, the old network connection, e.g. Internet connection, will be closed.

The Server NAPT (SNAPT) module according to the invention can be realized by means of appropriate software and/or hardware components. The SNAPT module comprises means to emulate the client application on the server side 24 and means to emulate the server application on the network side 25, e.g. Internet side (Figure 10). The SNAPT module creates for each server service 29, a server socket 26 on the Internet side. This server socket listens on the server service emulator port (server service emulator server socket). This port is different from the server service real port in order to avoid a bind error. For each server service request, the SNAPT module provides a client request emulation socket 60 on the server side 24. This socket relays packets to the real service server socket. For each client service the SNAPT module creates a server socket 28 on the server side 24. This server socket listens on the real client service port (client service emulator server socket). The client service emulator server sockets are grouped by CNAPT module ID. Those groups using the same ports are bound to different Virtual IP addresses 23 in order to avoid bind errors. For each client service request, the SNAPT module provides a server request emulation socket 61 on the Internet side 25. This socket relays packets to the right client service emulator server socket provided by the correspondent CNAPT ID. Further the SNAPT module comprises a control server socket 27 on the network side 25. The CNAPT module communicates with the SNAPT module over the network 30/31/32/33/34 through this "control server socket". This connection is used to exchange handshaking packets during an IP address-changing phase and it can be used to optimize the interaction between the client and the server application.

In the following will be described an embodiment according to the invention and the main embodiment variants. In each embodiment, shown here, a unique client application and a unique server application are shown in the appended figures. Nothing changes if there are n client applications 11 on the mobile device 10 that, through the local CNAPT module, are using the services provided by the m server applications 21 managed by the connected SNAPT module on the same machine 20. It is possible that each SNAPT module can accept the simultaneous connection from more than one CNAPT module.

In a first embodiment, the “Basic configuration”, the CNAPT module is installed on the client application mobile device 10 and the SNAPT module is installed on the same machine 20 as the server application 21 (Figure 11). The server IP address is set to “loopback” (50) in the client application. The CNAPT module sets the SNAPT IP address for each server service to ”ServerIP” (51). In this configuration the CNAPT module emulates the server services on the client side providing a server socket “loopback:ServiceEmulator” 16 for each of them (“loopback:ServiceA”, ”loopback:ServiceB” and so on). This socket can handle multiple concurrent client requests. The client application accesses the services by sending and receiving data to/from these sockets. On the server side the SNAPT module emulates the client application requests providing a socket “loopback:CasualPort” 60 for each of them. The Server application handles Client requests by sending and receiving data to/from these sockets.

In an embodiment variant, the “Multi-Site configuration”, the CNAPT module is installed on the client application mobile device 10 and it is connected simultaneously to more than one SNAPT module. Each SNAPT module is installed on the same machine as its server application (Figure 12). The Servers IP addresses are set to “loopback” (50) in the client application. The CNAPT module sets the SNAPT IP address to ”Server1IP” for the services A,B..Z and to “Server2IP” for the services 1,2..N (51). In Figure 12 a unique client application and a unique server application are shown. Nothing changes if there are n client applications on the mobile device that, through the local CNAPT module, are simultaneously using the services provided by the m server applications managed by all the connected SNAPT modules. This embodiment variant has the advantage that the system according to the invention can grant the seamless handover for the network connection among one or more client applications and two or more server applications running on different Internet nodes. With this embodiment a user 1, without interruption of service, can move from one local network provider to another one keeping active multiple sessions with multiple different service providers.

In an embodiment variant, the "Client PAN configuration" (PAN: Personal Area Network), the CNAPT module is installed on an additional mobile device 13, that could act as a "connectivity box" able to manage the highest

possible number of network access technologies and linked to the original mobile device through a broadband IP connection 131 (Figure 13). The client application mobile device 10 and the “connectivity box” have a fixed private IP address each, “ClientIP” and “CBoxIP”: they are part of a very small local network (e.g. they are connected using an Ethernet cross cable or a Wi-Fi ad-hoc connection or a Bluetooth connection). The server IP address is set to the “connectivity box” IP address (“CBoxIP”) (50) in the client application. The CNAPT module sets the SNAPT module IP address for each server service to ”ServerIP” (51). The CNAPT module comprises means to emulate the server services providing, on the client side, a server socket “CBoxIP: ServiceEmulator” 16 for each of them (“CBoxIP:ServiceA”, ” CBoxIP:ServiceB” and so on). This socket can handle multiple concurrent client requests. The client application accesses the services by sending and receiving data to/from these sockets. On the server side the SNAPT module comprises means to emulate the client application requests providing a socket “loopback:CasualPort” 60 for each of them. The server application handles client requests by sending and receiving data to/from these sockets. This embodiment variant has the advantage that the user 1 has to interact with a mobile device 10 simpler and maybe smaller than in the previous embodiments. In the previous embodiments this mobile device should have more than one network adapter and it must run the CNAPT module and the client application. In this embodiment variant instead it must have only one network adapter and it must run only the client application, while the system according to the invention, granting the seamless handover, is provided by the “connectivity box”. Another advantage of this embodiment variant is that, in order to avoid any modification of the client applications source code, it is not necessary to set the source IP address of the eventual client services to “loopback” address. In fact, in this case the source IP address is the client application device PAN IP address (“ClientIP”) 18. Since it does not change during a switch phase (only the CNAPT Internet IP address 19 changes), the client services will not be interrupted.

In an embodiment variant, the "Multi-Server LAN configuration", the CNAPT module is installed on the client application mobile device 10 and the SNAPT module is installed on an additional dedicated machine 70 placed in the

same network of the original server machine 20 (Figure 14); in this way a unique SNAPT module can manage more server applications on different machines at a time. The CNAPT module sets the SNAPT IP address for each server service to "SNAPTIP" (51). On the client side, the CNAPT module comprises means to emulate the servers' services providing a server socket "loopback:ServiceEmulator" 16 for each of them ("loopback:Service1A",.., "loopback:Service1Z,.., "loopback:ServiceNA",..., "loopback:ServiceNZ"). This socket can handle multiple concurrent client requests. The client application accesses the services by sending and receiving data to/from these sockets. The SNAPT module emulates the client application requests providing a socket "SNAPTIP:CasualPort" 60 for each of them. The server applications (1..N) handle client requests by sending and receiving data to/from these sockets. This embodiment variant has the advantage that a SNAPT module can manage simultaneously more different server applications running on different machines on the same network; in this way a company that provides more and differentiated server services can offer to its customers the seamless handover feature using a unique SNAPT machine.

In an embodiment variant, the "Multi-Server Internet configuration", the CNAPT module is installed on the client application mobile device 10 and the SNAPT module is installed on an additional dedicated Internet server 70 placed on an Internet site reachable from the CNAPT module and different from the Internet sites 20 running the server applications (Figure 15). This way the mobile device 10 is able to benefit by the above-mentioned advantages of the invention with any server providing services on the network, e.g. the Internet, without the need to have a SNAPT module installed on every single server of interest. The CNAPT module sets the SNAPT IP address for each server service to "SNAPTIP" (51). On the client side, the CNAPT module comprises means to emulate the Servers' services, providing a server socket "loopback:ServiceEmulator" 16 for each of them ("loopback:Service1A",.., "loopback:Service1Z,.., "loopback:ServiceNA" "loopback:ServiceNZ"). This socket can handle multiple concurrent Client requests. The Client application accesses the services by sending and receiving data to/from these sockets. The SNAPT module comprises means to emulate the client application requests providing a socket "SNAPTIP:CasualPort" 60 for each of them. The servers

applications (1..N) handle client requests by sending and receiving data to/from these sockets. This embodiment variant has the advantage that a SNAPT module, installed on an Internet node, can manage simultaneously more different server applications running on different Internet nodes. In this way the seamless handover of the network connection among one or more client applications and one or more server applications can be provided by a company, different from the services providers, that does not require any modification or installation on the services providers sites and that only requires the installation of the CNAPT module on the final customer mobile device. With such embodiment variant, following a switch between network access technology and/or provider, a customer of the above-described company (e.g. an active trader) would be able to maintain active different trading sessions with different trading service providers.

In an embodiment variant, the "Client PAN Multi-Server Internet configuration", the CNAPT module is installed on an additional mobile device, that could act as a “connectivity box” able to manage the highest possible number of network access technologies and linked to a set of mobile devices through a broadband IP connection (Figure 16). The client applications’ mobile devices and the “connectivity box” have a fixed private IP address each, “Client1IP”..”ClientNIP” and “CBoxIP”: they are part of a very small local network (e.g. they are connected using an Ethernet hub or a Wi-Fi ad-hoc connection or a Bluetooth connection). The Servers’ IP addresses are set to the “connectivity box” IP address (“CBoxIP”) in the client applications. The CNAPT module sets the SNAPT IP address for each server service to ”SNAPTIP” and sets the client IP address for each client service to the IP address of the mobile device running the related client application ("Client1IP" for client service 1A, 1B...1Z , ..., "ClientNIP" for client service NA, NB...NZ). The SNAPT module is installed on an additional dedicated network server or Internet server 70 placed on an Internet site reachable from the CNAPT module and different from the Internet sites 20 running the server applications. This way the mobile devices are able to benefit by the above-mentioned advantages of the invention with any server providing services on the network and/or Internet, without the need to have a SNAPT module installed on every single server of interest. The CNAPT module comprises means to emulate the servers’ services providing a

server socket “CBoxIP:ServiceEmulator” for each of them. This socket can handle multiple concurrent client requests. The client applications access the services by sending and receiving data to/from these sockets. The SNAPT module comprises means to emulate the client applications’ requests providing a socket “SNAPTIP:CasualPort” for each of them. The servers’ applications (1..N) handle client requests by sending and receiving data to/from these sockets. With such configuration, following a switch between network access technology and/or provider; for instance, an active trader would be able to maintain active multiple different sessions with multiple different service providers (e.g. a trading session with his laptop, a voice call with his VoiceIP Phone and a medical monitor session with his biometric sensors). This embodiment variant combines the advantages of the “Multi-Server Internet configuration” and the “Client PAN configuration”.

The set of embodiment variants described above is not intended to be exhaustive: it will be understood that further permutations of the base configurations and various changes in form and in detail may be made therein without departing from the spirit and the scope of the invention. The concept of the invention has been explained using a simplified model, but this concept is, obviously, more general: (1) The CNAPT module can emulate a great number of server and client services and can handle multiple concurrent requests to the same client or server service or to different services. (2) The CNAPT module can handle any number of client applications resident on the same device or in different devices. (3) The SNAPT module can handle multiple concurrent client requests coming from the same CNAPT module as well as multiple concurrent client requests coming from different CNAPT modules. (4) The SNAPT module can handle multiple concurrent server requests of client services coming from the same server application as well as multiple concurrent server requests coming from different server applications. (5) The SNAPT module can handle any number of server applications resident on the same device or in different devices. etc...

In an embodiment variant, a system according with the invention can support the dynamic update of the QoS (Quality of Service) parameters of the client and the server applications following a switch between two different

network providers. In fact the switch between two different network providers could sometimes determine a relevant variation in the bandwidth available or in others connection parameters like packet delay, packet loss probability etc. This variation should entail a QoS (Quality of Service) variation whenever the client and the server applications could dynamically change it. For instance, if the client and the server applications provide a real-time videoconference system, at their start a set of QoS parameters are exchanged and defined: frame rate, picture format, compression quality, ...etc. In case of bandwidth variation these applications should be able to adapt the QoS to the new conditions in order to provide the best possible service at all times. In such cases, the CNAPT module and the SNAPT module can be configured to send to the client and the server applications all the information they need to update their QoS.

In another embodiment variant, the client and/or server services byte streams between the CNAPT module and the SNAPT module could be compressed in order to reduce the amount of data exchanged. The CNAPT module and the SNAPT module, of course, have to know which services byte streams have to be compressed.

Description of the preferred embodiment

In the following an embodiment of the invention based on the "Basic configuration" will be described in more detail. For this example of an embodiment, it is assumed that the mobile device 10 using the client application 11 is able to hold two different network connections simultaneously, e.g. Internet connections, through the network adapter 1 and 2 (40 and 41 in Figure 17). The network adapter 1 and 2 are wireless adapters. In this example the server application 21 provides only two connection-oriented services (like services based on TCP): ServiceA and ServiceB (this assumption is used only to simplify the drawings; obviously the switch procedure handles also the connectionless server services). For each of them a server socket 29 listens for client application requests (Figure 17). The server machine is connected to Internet 34 and it has the IP address "ServerIP". The mobile device 10, using the wireless adapter 1 [40], is connected to Internet 34 via the wireless provider 1 [30] and it has the IP address "ClientIP_1". The client application does not

provide any client services (this assumption is used only to simplify the drawings; obviously the switch procedure handles also the connection-oriented/connectionless client services). The CNAPT module and the SNAPT module will be described as middleware software application installed respectively on the mobile device 10 and on the server PC 20. In a modified embodiment, the CNAPT module and the SNAPT module run on a dedicated device. In this case the client and the server applications send/receive packets to/from the external dedicated device. The unique difference from the other implementation is that the CNAPT module and SNAPT module are not referred to by the loopback address, but by their external dedicated device IP address.

The Client NAPT (CNAPT) module is a software and/or hardware module that comprises means to emulate the server application on the client side 14 and means to emulate the client application on the Internet side 15 (Figure 18). With only the server IP, the services' real ports, the services' emulator ports and the services' type, it provides: (1) for each server service, a server socket 16 on the client side. This server socket listens on the real service port (Server Service Emulator server socket). (2) For each server service request, a Client Request Emulation Socket 121 on the Internet side. This socket relays packets to the right Service Emulator Server Socket provided by the SNAPT module. To understand the functionality of the CNAPT module, suppose that the mobile device 10 is connected to Internet via the wireless provider 1 and has the IP address "ClientIP_1". If the server IP address has been set to "loopback" in the client application, when it requests the ServiceA, it sends a request to the "loopback:ServiceA" address 50. This request is received from the ServiceA Emulator server socket 16. The CNAPT module changes the source address from "loopback:ReqPort" 52 to "ClientIP_1:CasualPortX" 53 and the destination address from "loopback:ServiceA" 50 to "ServerIP:ServiceAEmulator" 54 and it then resends this request through the Client Request Emulator socket 121. When the Client Request Emulator socket 121 receives the server response, the CNAPT module changes the source address from "ServerIP:ServiceAEmulator" 55 to "loopback:ServiceA" 57 and the destination address from "ClientIP_1: CasualPortX" 56 to "loopback:ReqPort" 58 and it then resends this response through the Server Service Emulator server socket 16. The CNAPT module and

the Server NAPT (SNAPT) communicate through a "Control socket" 122. This connection is used to exchange handshaking packets during an IP address-changing phase and to optimize the interaction between the client and the server application. This connection is not always on; it is established when the CNAPT module has some needs, and it is closed when the operation has been completed.

If the client application provides some client services, they can also be emulated by the CNAPT module, but it has to know the client services' ports and type (connection-oriented or connectionless). With this information, the CNAPT module provides: (1) For each client service, a server socket 17 on the Internet side (Figure 8). This server socket listens on the client service emulator port (Client Service Emulator server socket). This port is different from the real client service port in order to avoid a bind error. (2) For each client service request, a Server Request Emulation Socket 123 on the client side. This socket relays packets to the real client server socket 111.

It is conceivable for a CNAPT module to emulate more than one server application simultaneously and in parallel. This CNAPT module can be used by more than one client application.

The CNAPT module preferably has to be started before the client application. It has to run until the client application stops in order to provide it with a good, reliable and cost effective network connection to the server application. The CNAPT module, at the start, does the following: (1) Obtains a network connection (Figure 20). If this is not possible, the CNAPT module can be stopped, for instance, and the client application cannot be started. (2) Launches the "decision task" module. (3) Creates the Server Service Emulator server sockets 16 on the Client side 14. (4) Creates the Client Service Emulator server sockets 17 on the network side 15. (5) Creates a local Control socket 122 connected to the SNAPT (or, in turn, to the SNAPTs in case of "Multi-Site configuration") Control server socket 27 and uses it to send a "CONNECT" packet from which the SNAPT can retrieve the CNAPT ID, its current IP address and the connection optimization parameters. When the SNAPT module receives the initial "CONNECT" packet, it does the following: (1) Checks

whether it can handle the new connection (obviously it has limited resources). (2) If it cannot accept the new connection, it replies with a "REFUSED" packet. (3) If it can accept a new connection, it stores the CNAPT ID and its current IP address. From now on, for this CNAPT ID, the SNAPT module will send all outgoing packets only to its current IP address. (3a) Allocates the internal resources for the new connection and adds the new CNAPT ID to the list of connected CNAPT modules. (3b) Creates on the Server side 24 the Client service emulator server sockets 28 related to the CNAPT ID and binds them to an available Virtual IP address 23. (3c) Sends the connect acknowledgement to the CNAPT module. The CNAPT module, at the end, has to create a local Control socket 122 connected to the SNAPT (or, in turn, to the SNAPTs in case of "Multi-Site configuration") Control server socket 27 and use it to send a "DISCONNECT" packet. When the SNAPT module receives a "DISCONNECT" packet it does the following: (1) Releases the internal resources of the disconnecting CNAPT ID and removes it from the list of connected CNAPT modules. (2) Destroys the Client service emulator server sockets 28 related to the CNAPT ID and unbinds them from the allocated Virtual IP address 23. (3) Sends the disconnect acknowledgement to CNAPT. The SNAPT module associates with each connected CNAPT module a maximum inactivity time (inactivity timeout). When this inactivity time has elapsed, the CNAPT will be considered disconnected, and the previous steps (1-3) will be executed.

The Server NAPT (SNAPT) module can be produced as a software and/or hardware module that emulates the client application on the server side 24 and the server application on the network or Internet side 25 (Figure 21). It provides: (1) For each server service, a server socket 26 on the Internet side 25. This server socket listens on the server service emulator port (Server Service Emulator server socket). This port is different from the real server service port in order to avoid a bind error. (b) For each service request, a Client Request Emulation Socket 60 on the server side 24. This socket relays packets to the real Service server socket 29. When the SNAPT module receives a ServiceA request, it changes the source address from "ClientIP_1:CasualPortX" 62 to "loopback:CasualPortY" 64 and the destination address from "ServerIP:ServiceAEmulator" 63 to "ServerIP:ServiceA" 65. It then resends this request through the Client Request Emulator socket 60. When the Client

Request Emulator socket receives the server application response, the SNAPT changes the source address from "ServerIP:ServiceA" 66 to "ServerIP:ServiceAEmulator" 68 and the destination address from "loopback:CasualPortY" 67 to "ClientIP_1:CasualPortX" 69. It then resends this response through the Server Service Emulator server socket 26. If the client applications provide some client services, they can also be emulated by the SNAPT module but it has to know the client services' ports and type (connection-oriented or connectionless), the CNAPT module services' emulator ports and for each client service the corresponding CNAPT module ID. With this information, the SNAPT module provides: (1) For each Client Service, a server socket 28 on the Server side. This server socket listens on the real Client Service port (Client Service Emulator server socket). (2) For each Client Service request, a Server Request Emulation Socket 61 on the network side, e.g. Internet side. This socket relays packets to the right Client Service Emulator Server Socket provided by the correspondent CNAPT module ID.

The SNAPT module can comprise means to emulate more than one client application request simultaneously and in parallel, and the SNAPT module can be used by more than one server application.

Suppose that the mobile device 10 is connected to the Internet via the Wireless Provider 1 [30], and it has the IP address "ClientIP_1". Suppose also that the client application 11 has already requested the ServiceA, and it is now exchanging data with the server application 21 (Figure 22). In this normal condition the "Control Socket" 122 is used only to exchange data for the Client/Server connection optimization. In the following, to explain the switch procedure, two cases will be illustrated: first the Wireless Provider 1 [30] that gradually becomes unavailable (slow switch) and then the Wireless Provider 1 that suddenly becomes unavailable (unexpected switch).

Now suppose that the Wireless Provider 1 (30) gradually becomes unavailable (slow switch), for instance the mobile device 10 is slowly leaving its coverage area, while the Wireless Provider 2 (31) remains still available. In order to keep the client application and the session active, a new Internet connection through the Provider 2 must be established before the Provider 1

becomes totally unavailable. If there is enough time for the IP transition phase (that is to say the Client and the Server applications don't go into timeout), the switch from the Wireless Provider 1 ("ClientIP_1") and the Wireless Provider 2 ("ClientIP_2") is totally transparent to the client and server applications that continue their execution without any interruption of service and/or session. To begin the IP transition phase from the Wireless Provider 1 ("ClientIP_1") to the Wireless Provider 2 ("ClientIP_2"), the CNAPT module has to (note that to begin this phase it is not necessary to have the "ClientIP_2" IP address, the unique precondition is to know the wireless provider selected to retrieve the new IP, in this case the Wireless Provider 2): (CNAPT-1) Suspend the "decision task" module activities (Search and Check activities). (CNAPT-2) Stop forwarding the connection-oriented server services request packets to the SNAPT module. (CNAPT-3) Stop forwarding the connection-oriented client services reply packets to the SNAPT module. (CNAPT-4) Stop forwarding the connectionless server services packets to the SNAPT module. (CNAPT-5) Buffer the pending server service requests and the pending client service replies. They will be forwarded at the end of the IP transition phase. (CNAPT-6) Flush all the transmission buffers. (CNAPT-7) Wait until the Search/Check activities have been suspended. (CNAPT-8) Put the input stream of the connection-oriented sockets linked to the SNAPT module in "waiting for last packet before redirection" mode. These sockets continue to receive data from the SNAPT module as before, until they receive a "LAST_MESSAGE_BEFORE_REDIRECTION" packet (note that this step can be avoided for connectionless services because they can tolerate packet losses). (CNAPT-9) Create a new Control socket 122 "ClientIP_1:CasualPortK" and connect it to the Control server socket 27 provided by SNAPT module on "ServerIP:ControlPort". (CNAPT-10) Send a "SWITCH_STEP1" packet to the SNAPT's Control server socket 27 using the Wireless Adapter 1 (Figure 23). This packet contains the CNAPT ID. (CNAPT-11) Wait for "SWITCH_STEP1" packet reception confirmation from the SNAPT module.

When the SNAPT module receives a "SWITCH_STEP1" packet, it has to (Figure 24): (SNAPT-1) Retrieve from this packet the ID of the sender CNAPT (from now on referred as "SWITCHING-CNAPT"). (SNAPT-2) Stop forwarding the connection-oriented server service reply packets to the

"SWITCHING-CNAPT". (SNAPT-3) Stop forwarding the connection-oriented client service request packets to the "SWITCHING-CNAPT". (SNAPT-4) Stop forwarding the connectionless client service packets to the "SWITCHING-CNAPT". (SNAPT-5) Buffer the pending server service replies and the pending client service requests. They will be forwarded at the end of the IP transition phase. (SNAPT-6) Flush all the transmission buffers. (SNAPT-7) Put the input stream of the connection-oriented sockets linked to the "SWITCHING-CNAPT" in "waiting for last packet before redirection" mode. These sockets continue to receive data from the "SWITCHING-CNAPT" as before until they receive a "LAST_MESSAGE_BEFORE_ REDIRECTION" packet (note that this step can be avoided for connectionless services because they can tolerate packet losses). (SNAPT-8) Send a "LAST_MESSAGE_BEFORE_REDIRECTION" packet in each output stream of the connection-oriented sockets linked to the "SWITCHING-CNAPT. (SNAPT-9) Confirm to the "SWITCHING-CNAPT" control socket 122 the reception of the "SWITCH_STEP1" packet with an "ACK" packet. (SNAPT-10) Wait until all the input streams in "waiting for last packet before redirection" mode have received the "LAST_MESSAGE_BEFORE_REDIRECTION" packet.

When the CNAPT module receives the SNAPT module acknowledgement, it has to (Figure 25): (CNAPT-12) Send a "LAST_MESSAGE_BEFORE_REDIRECTION" packet in each output stream of the connection-oriented sockets linked to the SNAPT. (CNAPT-13) Wait until all the input streams in "waiting for last packet before redirection" mode have received the "LAST_MESSAGE_BEFORE_REDIRECTION" packet. (CNAPT-14) Wait for SNAPT's "OK_TO_REDIRECTION" packet.

When the SNAPT module has received all the "LAST_MESSAGE_BEFORE_REDIRECTION" packets, it has to (Figure 26): (SNAPT-11) Send an "OK_TO_REDIRECTION" packet to the "SWITCHING-CNAPT" control socket 122. (SNAPT-12) Wait for the "CLIENT_SERVICES_READY_FOR_REDIRECTION" packet coming from the "SWITCHING-CNAPT" control socket 122.

When the CNAPT module receives the "OK_TO_REDIRECTION " packet, it has to (Figure 27): (CNAPT-15) Destroy the connectionless Client Services Emulator sockets 17 bound to the current CNAPT IP address and receiving data from the SNAPT. (CNAPT-16) Preserving the Client services connections between the Client and the Server applications, destroy the Client services connection-oriented emulation sockets linked to the SNAPT and generated by the connection-oriented Client Services Emulator server sockets 17. They were bound to the current CNAPT IP address. Each of these sockets has to be renewed by the SNAPT in order to preserve the Client services connections between the Client and the Server applications. (CNAPT-17) Preserving the Client services connections between the Client and the Server applications, destroy the connection-oriented Client Services Emulator server sockets 17 accepting requests from the SNAPT. They were bound to the current CNAPT IP address. (CNAPT-18) Send a "CLIENT_SERVICES_READY_FOR_REDIRECTION" packet to the SNAPT's control server socket 27. (CNAPT-19) Wait for the "SERVER_SERVICES_READY_FOR_REDIRECTION" packet coming from the SNAPT's control server socket 27.

When the SNAPT module receives the "CLIENT_SERVICES_READY_FOR_REDIRECTION" packet, it has to (Figure 28): (SNAPT-13) Preserving the Client services connections between the Client and the Server applications, destroy the connection-oriented Client Services Server Request Emulation sockets 61 linked to the current "SWITCHING-CNAPT" IP address. (SNAPT-14) Preserving the Server services connections between the Client and the Server applications, destroy the Server services connection-oriented emulation sockets linked to the current "SWITCHING-CNAPT" IP address and generated by the connection-oriented Server Services Emulator server sockets 26. Each of these sockets has to be renewed by the "SWITCHING-CNAPT" in order to preserve the Server services connections between the Client and the Server applications. (SNAPT-15) Send a "SERVER_SERVICES_READY_FOR_REDIRECTION" packet to the "SWITCHING-CNAPT" control socket 122. (SNAPT-16) Wait for the "SWITCH_STEP2" packet coming from the "SWITCHING-CNAPT" through the new control socket 122 bound to its new IP address.

When the CNAPT module receives the "SERVER_SERVICES_READY_FOR_REDIRECTION " packet, it has to (Figure 29): (CNAPT-20) Preserving the Server services connections between the Client and the Server applications, destroy the connection-oriented Server services Client Request Emulation sockets 121 bound to the current "SWITCHING-CNAPT" IP address. (CNAPT-21) Destroy the control socket 122 bound to the current "SWITCHING-CNAPT" IP address. (CNAPT-22) If needed, close the old Internet connection through the Wireless Provider 1 (if the client application and the server application are connected through a VPN, this step has to be preceded by the following step: Close the existing VPN connection between "ClientIP_1" and "ServerIP"). (CNAPT-23) If the "ClientIP_2" address is not yet available, open the new Internet connection through the selected wireless adapter (in this case the Wireless Adapter 2). (CNAPT-24) Change the current IP address from the old IP address to the new one (if the client application and the server application are connected through a VPN, this step has to be followed by the following step: Open the new VPN between "ClientIP_2" and "ServerIP"). (CNAPT-25) Change to the new CNAPT IP address the binding of the server services connectionless sockets transmitting to the SNAPT module. (CNAPT-26) Recreate the connectionless Client Services Emulator sockets 17 receiving data from the SNAPT and bind them to the new CNAPT IP address. (CNAPT-27) Preserving the Client services connections between the Client and the Server applications, recreate the connection-oriented Client Services Emulator server sockets 17 accepting requests from the SNAPT and bind them to the new CNAPT IP address. (CNAPT-28) Create a new control socket 122 "ClientIP_2:CasualPort" and connect it to the control server socket 27 provided by SNAPT module on "ServerIP:ControlPort". (CNAPT-29) Send a "SWITCH_STEP2" packet to the SNAPT's control server socket 27 using the Wireless Adapter 2. This packet contains the CNAPT ID and through this packet the SNAPT module can deduce the new CNAPT IP address. (CNAPT-30) Wait for the renewal of the connection-oriented Client services emulation sockets destroyed in [CNAPT-16]. This renewal will be done with a connection request [SNAPT-19] of the SNAPT for each connection-oriented socket generated by the Client Services Emulator server sockets 17.

When the SNAPT module receives the "SWITCH_STEP2" packet, it has to: (SNAPT-17) Update the "SWITCHING-CNAPT" IP address with its new IP address as retrieved by the source field of the "SWITCH_STEP2" control packet. (SNAPT-18) Redirect the Client services connectionless sockets transmitting to the "SWITCHING-CNAPT" from its old IP address to the new one (no redirection is needed for the connectionless server sockets receiving data from the "SWITCHING-CNAPT"; they are unaffected by the CNAPT IP address switch). (SNAPT-19) Recreate the connection-oriented Client Services Server Request Emulation sockets 61 destroyed in [SNAPT-13] and connect them, in order to preserve the Client/Server interaction, to the Client Services Emulator server sockets 17 provided on the new "SWITCHING-CNAPT" IP address. (SNAPT-20) Wait for the renewal of the connection-oriented Server services emulation sockets destroyed in [SNAPT-14]. This renewal will be done with a connection request [CNAPT-32] of the "SWITCHING_CNAPT" for each connection-oriented socket generated by the Server Services Emulator server sockets 26.

When the CNAPT module has received the renewal of the connection-oriented Client services emulation sockets destroyed in [CNAPT-16] it has to (Figure 30): (CNAPT-31) Redirect the correspondent Client services emulation connections to the new IP address through the renewed emulation sockets in order to preserve the Client/Server interaction. (CNAPT-32) Recreate the connection-oriented Server services Client Request Emulation sockets 121 destroyed in [CNAPT-20], bind them to the new "SWITCHING-CNAPT" IP address and connect them, in order to preserve the Client/Server interaction, to the Server Services Emulator server sockets 26 provided on the SNAPT IP address. (CNAPT-33) Wait for the "ALL_REDIRECTED" packet coming from the SNAPT's control server socket 27.

When the SNAPT module receives the renewal of the connection-oriented Server services emulation sockets destroyed in [SNAPT-14] it has to (Figure 31): (SNAPT-21) Redirect the correspondent Server services emulation connections to the new "SWITCHING-CNAPT" IP address through the renewed emulation sockets in order to preserve the Client/Server interaction. (SNAPT-22) Send an "ALL_REDIRECTED" packet to the new "SWITCHING-CNAPT"

control socket 122. (SNAPT-23) Wake up all the connectionless/connection-oriented client and server services and send the packets buffered in [SNAPT-5].

When the CNAPT module receives the "ALL_REDIRECTED" packet, it has to (Figure 32): (CNAPT-34) Wake up all the connectionless/connection-oriented client and server services and the "decision task" activities (Search and Check activities) and send the packets buffered in [CNAPT-5].

In the steps [CNAPT-11], [CNAPT-13], [CNAPT-14], [CNAPT-19], [CNAPT-30] and [CNAPT-33] the CNAPT waits for at most Z seconds (switch timeout). When this time has elapsed the CNAPT and the related Client application should be stopped. In the steps [SNAPT-10], [SNAPT-12], [SNAPT-16] and [SNAPT-20] the SNAPT waits for at most Z seconds (switch timeout). When this time has elapsed the SNAPT could consider the involved CNAPT as disconnected and it should execute these steps: (1) Release the internal resources of the disconnected CNAPT ID and remove it from the list of connected CNAPT. (2) Destroy the Client service emulator server sockets related to the CNAPT ID and unbind them from the allocated Virtual IP address.

In case of "Multi-Site configuration", some groups of steps are executed separately for each connected SNAPT and there are some synchronization steps. The steps CNAPT-9..13 are executed separately for each connected SNAPT. The step CNAPT-14 is a synchronization step for all the connected SNAPTs. When the step CNAPT-18 has been completed for all the connected SNAPTs, the steps CNAPT-19..21 are executed separately for each of them. The steps CNAPT-28,29 are executed separately for each connected SNAPT. The steps CNAPT-32,33 are executed separately for each connected SNAPT.

Suppose now that the Wireless Provider 1 becomes suddenly unavailable (unexpected switch), for instance the mobile device is rapidly leaving its coverage area, while the Wireless Provider 2 remains still available. In order to maintain active the client application and the session, a new Internet connection through the Provider 2 must be established. If there is enough time for the IP transition phase (that is to say the Client and the Server applications

don't go into timeout), the switch from the Wireless Provider 1 ("ClientIP_1") and the Wireless Provider 2 ("ClientIP_2") is totally transparent to the Client and Server applications that continue their execution without any interruption of service and/or session. Note that the IP transition phase can be generated also by a temporary interruption of the Internet connection provided by the Wireless Provider 1, that however remain still available; in fact that temporary interruption may cause a modification on the assigned IP, e.g. from "ClientIP_1" to "ClientIP_1a".

If the Wireless Provider 1 becomes suddenly unavailable the correspondent Internet connection will be suddenly interrupted. This interruption will throw on the CNAPT software some exceptions that are used to handle this particular event if and only if the CNAPT was reading or writing on the connection at the time of the interruption; otherwise the Check Activity signals the interruption and invokes the IP transition phase. The interruption will throw on the SNAPT software some exceptions that are used to handle this particular event if and only if the SNAPT was reading or writing on the connection to that CNAPT ID at the time of the interruption; otherwise the interruption is signalled by the reception of the "SWITCH_UNEXPECTED" packet from that CNAPT ID. The reception of the "SWITCH_UNEXPECTED" packet causes the invocation of the IP transition phase for that CNAPT ID.

When the CNAPT catches the connection exception or when the Check Activity signals the interruption, the CNAPT has to establish a new Internet connection through the Wireless Provider 2 (or, in turn, through the Wireless Provider 1 if it is became available in the meanwhile, e.g. there was only a temporary problem). When the new Internet connection is available, to begin the IP transition phase from the "ClientIP_1" (no more available) to the "ClientIP_2" the CNAPT has to: (CNAPT-1*) Suspend the "decision task" activities (Search and Check activities). (CNAPT-2*) Stop forwarding the connection-oriented Server services request packets to the SNAPT. (CNAPT-3*) Stop forwarding the connection-oriented Client services reply packets to the SNAPT. (CNAPT-4*) Stop forwarding the connectionless Server services packets to the SNAPT. (CNAPT-5*) Buffer the pending Server service requests and the pending Client service replies. They will be forwarded at the end of the

IP transition phase. (CNAPT-6*) For each outgoing connection, store the eventual unsent packets (i.e. the packets that the outgoing connection was eventually sending when the interruption exception was caught). (CNAPT-7*) Wait until the Search/Check activities have been suspended. (CNAPT-8*) Destroy the connectionless Client Services Emulator sockets 17 bound to the old CNAPT IP address. (CNAPT-9*) Preserving the Client services connections between the Client and the Server applications, destroy the Client services connection-oriented emulation sockets linked to the SNAPT and generated by the connection-oriented Client Services Emulator server sockets 17. They were bound to the old CNAPT IP address. Each of these sockets has to be renewed by the SNAPT in order to preserve the Client services connections between the Client and the Server applications. (CNAPT-10*) Preserving the Client services connections between the Client and the Server applications, destroy the connection-oriented Client Services Emulator server sockets 17 accepting requests from the SNAPT. They were bound to the old CNAPT IP address. (CNAPT-11*) Preserving the Server services connections between the Client and the Server applications, destroy the connection-oriented Server services Client Request Emulation sockets 121 bound to the old CNAPT IP address. (CNAPT-12*) Change the current IP address from the old IP address to the new one (if the client application and the server application are connected through a VPN, this step has to be followed by the following step: Open the new VPN between "ClientIP_2" and "ServerIP"). (CNAPT-13*) Change to the new CNAPT IP address the binding of the Server services connectionless sockets transmitting to the SNAPT. (CNAPT-14*) Recreate the connectionless Client Services Emulator sockets 17 receiving data from the SNAPT and bind them to the new CNAPT IP address. (CNAPT-15*) Recreate the connection-oriented Client Services Emulator server sockets 17 accepting requests from the SNAPT and bind them to the new CNAPT IP address. (CNAPT-16*) Create a new Control socket "ClientIP_2:CasualPortK" and connect it to the Control server socket provided by SNAPT on "ServerIP:ControlPort". (CNAPT-17*) Send a "SWITCH_UNEXPECTED" packet to the SNAPT "Control server socket" using the new Internet connection. This packet contains the CNAPT ID and through this packet the SNAPT can deduce the new CNAPT IP address. (CNAPT-18*) Wait for the renewal of the connection-oriented Client services emulation sockets destroyed in [CNAPT-9*]. This renewal will be done with a connection

request [SNAPT-11*] of the SNAPT for each connection-oriented socket generated by the Client Services Emulator server sockets 17.

When the SNAPT catches the connection exception it waits for a "SWITCH_UNEXPECTED" packet coming from that CNAPT ID. The SNAPT associates to that CNAPT ID a max reconnection waiting time (switch timeout). When this reconnection waiting time has elapsed, the related CNAPT will be considered definitely disconnected and the SNAPT should execute these steps: (1) release the internal resources of the disconnected CNAPT ID and remove it from the list of connected CNAPT; (2) destroy the Client service emulator server sockets related to the CNAPT ID and unbind them from the allocated Virtual IP address. When, in any way, the SNAPT receives a "SWITCH_UNEXPECTED" packet it has to: (SNAPT-1*) Retrieve from this packet the ID of the sender CNAPT (from now on referred as "SWITCHING-CNAPT"). Through this packet the SNAPT can also deduce the new "SWITCHING-CNAPT" IP address. (SNAPT-2*) Stop forwarding the connection-oriented Server service reply packets to the "SWITCHING-CNAPT". (SNAPT-3*) Stop forwarding the connection-oriented Client service request packets to the "SWITCHING-CNAPT". (SNAPT-4*) Stop forwarding the connectionless Client service packets to the "SWITCHING-CNAPT". (SNAPT-5*) Buffer the pending Server service replies and the pending Client service requests. They will be forwarded at the end of the IP transition phase. (SNAPT-6*) For each outgoing connection, store the eventual unsent packets (i.e. the packets that the outgoing connection was sending when the interruption exception was caught). (SNAPT-7*) Preserving the Client services connections between the Client and the Server applications, destroy the connection-oriented Client Services Server Request Emulation sockets 61 linked to the old "SWITCHING-CNAPT" IP address. (SNAPT-8*) Preserving the Server services connections between the Client and the Server applications, destroy the Server services connection-oriented emulation sockets linked to the old "SWITCHING-CNAPT" IP address and generated by the connection-oriented Server Services Emulator server sockets 26. Each of these sockets has to be renewed by the "SWITCHING-CNAPT" in order to preserve the Server services connections between the Client and the Server applications. (SNAPT-9*) Update the "SWITCHING-CNAPT" IP address with its new IP address as retrieved by the source address field of the

"SWITCH_UNEXPECTED" control packet. (SNAPT-10*) Redirect the Client services connectionless sockets transmitting to the "SWITCHING-CNAPT" from its old IP address to the new one (No redirection is needed for the connectionless Server services sockets receiving data from the "SWITCHING-CNAPT". They are unaffected by the CNAPT IP address switch.). (SNAPT-11*) Recreate the connection-oriented Client Services Server Request Emulation sockets 61 destroyed in [SNAPT-7*] and connect them, in order to preserve the Client/Server interaction, to the Client Services Emulator server sockets 17 provided on the new "SWITCHING-CNAPT" IP address. (SNAPT-12*) Wait for the renewal of the connection-oriented Server services emulation sockets destroyed in [SNAPT-8*]. This renewal will be done with a connection request [CNAPT-21*] of the "SWITCHING_CNAPT" for each connection-oriented socket generated by the Server Services Emulator server sockets 26.

When the CNAPT has received the renewal of the connection-oriented Client services emulation sockets destroyed in [CNAPT-9*] it has to: (CNAPT-19*) Redirect the correspondent Client services emulation connections to the new IP address through the renewed emulation sockets in order to preserve the Client/Server interaction. (CNAPT-20*) Recreate the connection-oriented Server services Client Request Emulation sockets 121 destroyed in [CNAPT-11*], bind them to the new "SWITCHING-CNAPT" IP address and connect them, in order to preserve the Client/Server interaction, to the Server Services Emulator server sockets 26 provided on the SNAPT IP address. (CNAPT-21*) Wait for the "ALL_REDIRECTED" packet coming from the SNAPT "Control server socket".

When the SNAPT receives the renewal of the connection-oriented Server services emulation sockets destroyed in [SNAPT-8*] it has to: (SNAPT-13*) Redirect the correspondent Server services emulation connections to the new "SWITCHING-CNAPT" IP address through the renewed emulation sockets in order to preserve the Client/Server interaction. (SNAPT-14*) Send an "ALL_REDIRECTED" packet to the new "SWITCHING-CNAPT" control socket. (SNAPT-15*) Resend the unsent packets eventually stored in [SNAPT-6*]. (SNAPT-16*) Wake up all the connectionless/connection-oriented Client and Server services and send the packets buffered in [SNAPT-5*].

When the CNAPT receives the "ALL_REDIRECTED" packet it has to: (CNAPT-22*) Resend the unsent packets eventually stored in [CNAPT-6*]. (CNAPT-23*) Wake up all the connectionless/connection-oriented Client and Server services and the "decision task" activities (Search and Check activities) and send the packets buffered in [CNAPT-5*].

In the steps [CNAPT-18*] and [CNAPT-21*] the CNAPT waits for at most Z seconds (switch timeout). When this time has elapsed the CNAPT and the related Client application should be stopped. When the SNAPT is waiting for the "SWITCH_UNEXPECTED" packet coming from a switching CNAPT and in the step [SNAPT-12*], the SNAPT waits for at most Z seconds (switch timeout). When this time has elapsed the SNAPT could consider the involved CNAPT as disconnected and it should execute these steps: (1) Release the internal resources of the disconnected CNAPT ID and remove it from the list of connected CNAPT. (2) Destroy the Client service emulator server sockets related to the CNAPT ID and unbind them from the allocated Virtual IP address.

In case of "Multi-Site configuration", some groups of steps are executed separately for each connected SNAPT and there are some synchronization steps. The steps CNAPT-16*/17* are executed separately for each connected SNAPT. The step CNAPT-18* is a synchronization step for all the connected SNAPTs. The steps CNAPT-20*/21* are executed separately for each connected SNAPT.

It is obvious that the minimal duration of the IP transition phase, therefore the connection timeout of the client and the server application, must be higher than the time requested for the SNAPT module and CNAPT module activities described above. At the end of the IP transition phase, the CNAPT module and the SNAPT module, if suitably configured, could additionally send to the client/server applications special information, e.g. all the information they need to update their QoS (Quality of Service) etc. Furthermore it is clear that the above description is based on a set of assumptions for this specific example. However, the scope of the invention encompasses many different embodiments or examples for implementing different features of the invention. Therefore the described example is not intended to limit the invention in any

way. While the invention has been particularly shown and described with reference to the preferred embodiment, it will be understood by those skilled in the art that various changes in form and in detail may be made therein without departing from the spirit and the scope of the invention.

A system according to the invention, e.g. based on wireless technologies, can help organizations to improve and diversify their range of services. For instance that system can help the companies' mobile field forces to convey to their customers a sense of efficiency, competence and customer-centric "philosophy". Such system provides the User with the best Internet connection in term of bandwidth, reliability and cost effectiveness among all the wired/wireless network access technologies and/or access providers available at a certain time and location. Let's consider a User connected via GPRS to a competence centre using videoconference and shared whiteboard, i.e. a collaboration application. If the User is moving from an only GPRS covered area to a Wi-Fi covered area, for instance he is going to an airport, the system according to the invention can ask him if he wants to switch to the available Wi-Fi connection. If he accepts it can switch, automatically or with a limited user interaction, from the old GPRS connection to the new Wi-Fi connection. The switch is totally transparent for the collaboration application (no restart is needed). The used services remain up and running, and the User experiences a better QoS (as a matter of fact in this case the videoconference and shared whiteboard performance improves). Let's now consider the case where the same User leaves the Wi-Fi covered area. In this case the system can detect the Wi-Fi signal strength degradation, and before the Wi-Fi signal disappears it automatically makes a switch to a GPRS connection if the GPRS signal is present. Also in this case the switch is totally transparent for the collaboration application (no restart is needed). The User only experiences a worse QoS, but the services remain up and running. A system according to the invention requires only limited CPU and/or memory resources so it can be used with PDAs and smart-phones, not only with laptops. Operating at application level (L7 of the OSI-7 Layers Protocols Stack), that system can be easily and quickly adapted to the wireless world innovation and, moreover, it can use all the widespread commercial products at transport (L4) or network (L3) or below level (L1, L2) to provide all the security features. It doesn't require the implementation

of any custom security feature. Finally, the porting to the various operating systems (Windows NT-2000-CE-Mobile, Linux, Symbian, PalmOS, etc...) can be easily achieved by using Java technologies wherever it is possible.

Claims

1. Method for seamless handover of mobile devices in heterogeneous networks in which method a mobile device (10) is moved between different topological network locations (30/31/32/33) and/or transmits and/or receives data by means of different network access technologies without the data transfer of active Client IP applications being interrupted, characterised in

that a Client IP application (11) of the mobile device (10) makes an IP request to a client-service module (12) on the mobile device (10),

that the client-service module (12) creates a logical data tunnel with a server-service module (22) and reroutes the Client IP request to it, and

that the server-service module (22) reroutes the Client IP request to a Server IP application (21) that creates an IP socket for the Client IP request to handle the exchange of IP data packets with the Client IP application (11).

2. Method according to claim 1, characterised in that the client-service module (12) provides at least a server application emulation interface comprising IP sockets to be used by the Client IP application (11) and a client application emulation interface comprising IP sockets to be used by the server-service module (22), and

that the server-service module (22) provides at least a server application emulation interface comprising IP sockets to be used by the client-service module (12) and a client application emulation interface comprising IP sockets to be used by the Server IP application (21).

3. Method according to claim 2, characterised in that the server application emulation interface of the client-service module (12) comprises IP sockets bound to the loopback IP address and the client application emulation interface of the client-service module (12) comprises IP sockets bound to the mobile device IP address used by the client-service module (12).

4. Method according to one of the claims 2 or 3, characterised in

that the Client IP application (11) creates for the Client IP request a client application request socket connected to the client-service module (12),

that the client-service module (12) provides a server service emulator server socket accepting the Client IP request,

that the client-service module (12) creates a client request emulation socket connected to the server-service module (22) to reroute the Client IP request,

that the server-service module (22) provides a server service emulator server socket accepting the rerouted Client IP request of the client-service module (12),

that the server-service module (22) creates a client request emulation socket connected to the Server IP application (11) to reroute the Client IP request, and

that the Server IP application (11) provides a server service server socket accepting the Client IP request of the client application.

5. Method according to one of the claims 1 to 4, characterised in that the server-service module (22) runs on a network device (70) different to the network device (20) on which runs the Server IP application (21).

6. Method according to one of the claims 1 or 2 or 4 or 5, characterised in that the client-service module (12) runs on a first network device and the Client IP application (11) on a second different network device, whereas the first network device and the second network device are connected via a broadband network connection forming a logical mobile unit together.

7. Method according to claim 6, characterised in that the network connection consists of a broadband network connection comprising Bluetooth and/or Ethernet and/or Wi-Fi.

8. Method according to one of the claims 1 to 7, characterised in that the Server IP application (21) sends its IP response to the server-service module (22), wherein the Server application IP response path is the mirror image of the Client IP request path.

9. Method according to one of the claims 1 to 8, whereas a Server IP application (21) makes an IP request to the server-service module (22), characterised in

that the Server IP application (21) creates for the Server IP request a server application request socket connected to the server-service module (22),

that the server-service module (22) provides a client service emulator server socket accepting the Server IP request,

that the server-service module (22) creates a server request emulation socket connected to the client-service module (12) to reroute the Server IP request,

that the client-service module (12) provides a client service emulator server socket accepting the rerouted Server IP request of the server-service module (22),

that the client-service module (12) creates a server request emulation socket connected to the Client IP application (11) to reroute the Server IP request, and

that the Client IP application (11) provides a client service server socket accepting the Server IP request of the server application.

10. Method according to one of the claims 1 to 9, characterised in that the client-service module (12) and/or the server-service module (22) are pure Layer 7 application according to the Open System Interconnection Structure of the International Organization for Standardization.

11. Method according to claim 10, characterised in that the client-service module (12) and/or the server-service module (22) are created as cross platform modules.

12. Method according to claim 11, characterised in that the client-service module (12) and/or the server-service module (22) are composed at least in parts by Java modules.

13. Method according to one of the claims 1 to 12, characterised in that a plurality of client-service modules (12) of one or more mobile devices are assigned to the same server-service module (22) and the client application emulation interface sockets of the server-service module (22) are bound to different Virtual IP addresses created and/or allocated by it.

14. Method according to claims 1 to 13, characterised in that a plurality of Server IP applications (21) resident on one or more network devices are assigned to the same server-service module (22).

15. Method according to claims 1 or 2 or 4 to 14, characterised in that a plurality of Client IP applications (11) resident on one or more network devices are assigned to the same client-service module (12).

16. Method according to the claims 1 to 15, characterised in that the client-service module (12) is assigned simultaneously to a plurality of server-service modules (22) resident on one or more network devices.

17. Method according to claim 16, characterised in that for different services the client-service module (12) is assigned to the plurality of server-service modules (22), whereas each service is handled by a corresponding server-service module (22).

18. Method according to one of the claims 1 to 17, characterised in that the client-service module (12) checks the mobile device (10) for available physical network interfaces and creates a lookup table with the available and configurable physical network interfaces, the client-service module creating the data tunnel via one of the physical network interfaces contained in that lookup table.

19. Method according to claim 18, characterised in that the client-service module (12) of the mobile device (10) checks periodically for available physical network interfaces.

20. Method according to one of the claims 18 or 19, characterised in that, with a change or an update of the physical network interface of the mobile device used by the client-service module (12), the connection of the data tunnel to the network is kept up and updated by means of the client-service module (12) using the lookup table.

21. Method according to one of the claims 18 to 20, characterised in that the client-service module (12) automatically changes and updates the physical network interfaces of the mobile device on the basis of information from the lookup table.

22. Method according to one of the claims 18 to 21, characterised in that the client-service module (12) switches from an old IP address to a new IP address by means of a switching procedure which keeps up and updates the data tunnel exchanging IP control packets with the server-service module (22) in order to provide a seamless handover.

23. Method according to one of the claims 18 to 21, characterised in that the client-service module (12) switches suddenly from an old IP address no more available to a new IP address by means of a switching procedure which restores and updates the data tunnel exchanging IP control packets with the server-service module (22) in order to provide a seamless handover.

24. Method according to one of the claims 1 to 23, characterised in that the criteria for the automatic change and/or update of the physical network interface by the client-service module (12) are defined by the user.

25. Method according to the claims 1 to 24, characterised in that a change or an update of the physical network interface used by the client-service module (12) is initiated by the user.

26. Method according to one of the claims 1 to 25, characterised in that the available physical network interfaces are configured dynamically.

27. Method according to one of the claims 1 to 25, characterised in that the available physical network interfaces are configured statically.

28. Method according to one of the claims 1 to 27, characterised in that if the network connection used by the client-service module (12) is interrupted, IP data packets coming from the Client IP applications (11) are buffered in a data buffer of the client-service module (12) until the data tunnel to the server-service module (22) is restored.

29. Method according to one of the claims 1 to 28, characterised in that if the network connection to the client-service module (12) is interrupted, IP data packets coming from the Server IP applications (21) are buffered in a data buffer of the server-service module (22) until the data tunnel to the client-service module (12) is restored.

30. System for seamless handover of mobile devices in heterogeneous networks, in which a mobile device has at least one physical network interface and is movable between different topological network locations, and data are transmittable bidirectionally by means of different network access technologies without the data transfer of active Client IP applications being interrupted, characterised in

that the mobile device (10) comprises a client-service module (12),

that the client-service module (12) comprises means for creating a logical data tunnel with a server-service module (22),

that the client-service module (12) comprises means to reroute the IP data packets coming from a Client IP application (11) to the server-service module (22) and to reroute the IP data packets coming from the server-service module (22) to a Client IP application (11), and

that the server-service module (22) comprises means to reroute the IP data packets coming from a client-service module (12) to a Server IP application (21) and to reroute the IP data packets coming from a Server IP application (21) to a client-service module (12).

31. System according to claim 30, characterised in that the client-service module (12) comprises at least a server application emulation interface composed of IP sockets to be used by the Client IP application (11) and a client application emulation interface composed of IP sockets to be used by the server-service module (22), and

that the server-service module (22) comprises at least a server application emulation interface composed of IP sockets to be used by the client-service module (12) and a client application emulation interface composed of IP sockets to be used by the Server IP application (21).

32. System according to claim 31, characterised in that the server application emulation interface of the client-service module (12) is composed of IP sockets bound to the loopback IP address and the client application emulation interface of the client-service module (12) is composed of IP sockets bound to the mobile device IP address used by the client-service module (12).

33. System according to one of the claims 31 or 32, characterised in

that the Client IP application (11) comprises for the Client IP request a client application request socket connected to the client-service module (12),

that the client-service module (12) comprises a server service emulator server socket accepting the Client IP request,

that the client-service module (12) comprises a client request emulation socket connected to the server-service module (22) to reroute the Client IP request,

that the server-service module (22) comprises a server service emulator server socket accepting the rerouted Client IP request of the client-service module (12),

that the server-service module (22) comprises a client request emulation socket connected to the Server IP application (11) to reroute the Client IP request, and

that the Server IP application (11) comprises a server service server socket accepting the Client IP request of the client application.

34. System according to one of the claims 30 or 31 or 33, characterised in that the system comprises a first network device for the client-service module (12) and a second different network device for the Client IP application (11), whereas the first network device and the second network device are connected via a broadband network connection forming a logical mobile unit together.

35. System according to claim 34, characterised in that the network connection comprises a broadband network connection comprising Bluetooth and/or Ethernet and/or Wi-Fi.

36. System according to one of the claims 30 to 35, characterised in that the Server application IP response path is the mirror image of the Client IP request path.

37. System according to one of the claims 30 to 36, whereas in case of an IP request by a Server IP application (21) to the server-service module (22), characterised in

that the Server IP application (21) comprises for the Server IP request a server application request socket connected to the server-service module (22),

that the server-service module (22) comprises a client service emulator server socket accepting the Server IP request,

that the server-service module (22) comprises a server request emulation socket connected to the client-service module (12) to reroute the Server IP request,

that the client-service module (12) comprises a client service emulator server socket accepting the rerouted Server IP request of the server-service module (22),

that the client-service module (12) comprises a server request emulation socket connected to the Client IP application (11) to reroute the Server IP request, and

that the Client IP application (11) comprises a client service server socket accepting the Server IP request of the server application.

38. System according to one of the claims 30 to 37, characterised in that the client-service module (12) and/or the server-service module (22) are pure Layer 7 application according to the Open System Interconnection Structure of the International Organization for Standardization.

39. System according to claim 38, characterised in that the client-service module (12) and/or the server-service module (22) are as cross platform modules.

40. System according to claim 39, characterised in that the client service module (12) and/or the server-service module (22) are composed at least in parts by Java modules.

41. System according to one of the claims 30 to 40, characterised in that a plurality of client-service modules (12) of one or more mobile devices are assigned to the same server-service module (22) and the client application emulation interface sockets of the server-service module (22) are bound to different Virtual IP addresses created and/or allocated by it.

42. System according to claims 30 to 41, characterised in that a plurality of Server IP applications (21) resident on one or more network devices are assigned to the same server-service module (22).

43. System according to claims 30 and 31 and to claims 33 to 42, characterised in that a plurality of Client IP applications (11) resident on one or more network devices are assigned to the same client-service module (12).

44. System according to the claims 30 to 43, characterised in that the client-service module (12) is assigned simultaneously to a plurality of server-service modules (22) resident on one or more network devices.

45. System according to claim 44, characterised in that for different services the client-service module (12) is assigned to the plurality of server-service modules (22), whereas each service is handled by a corresponding server-service module (22).

46. A computer program product comprising a computer-readable medium with computer program code means contained therein for control of one or more processors of a computer-based system for seamless handover of mobile devices in heterogeneous networks, wherein

by means of the computer program product a client-service module (12) of a mobile device creates a logical data tunnel with a server-service

module (22) and reroutes to it the IP packets coming from a Client IP application (11), and

by means of the computer program product the server-service module (22) reroutes to the client-service module (12) the IP packets coming from the Server IP application (21).

47. A computer program product which is able to be loaded in the internal memory of a digital computer and comprises software code sections with which the steps according to one of the claims 1 to 29 are able to be carried out when the product runs on a computer.

Abstract

The invention relates to a method and system for seamless handover of mobile devices in heterogeneous networks. A mobile device (10) is moved between different topological network locations (30/31/32/33) and/or transmits and/or receives data by means of different network access technologies without the data transfer of the active Client IP applications being interrupted, whereas a Client IP application (11) of the mobile device (10) makes an IP request to a client-service module (12) on the mobile device (10), whereas the client-service module (12) creates a logical data tunnel with a server-service module (22) and reroutes the Client IP request to it and whereas the server-service module (22) reroutes the Client IP request to a Server IP application (21) that creates an IP socket for the Client IP request to handle the exchange of IP data packets with the Client IP application (11).

(Figure 11)

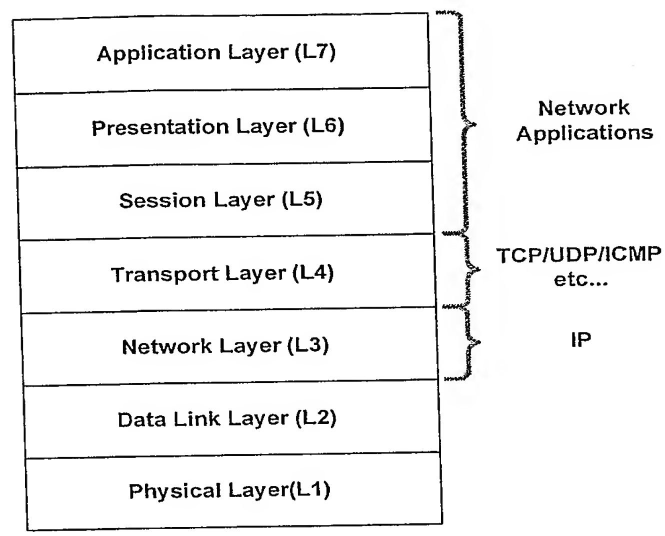

Application Layer (L7)
Presentation Layer (L6)
Session Layer (L5)
Transport Layer (L4)
Network Layer (L3)
Data Link Layer (L2)
Physical Layer(L1)
Network Applications
TCP/UDP/ICMP etc...
IP

FIG.1

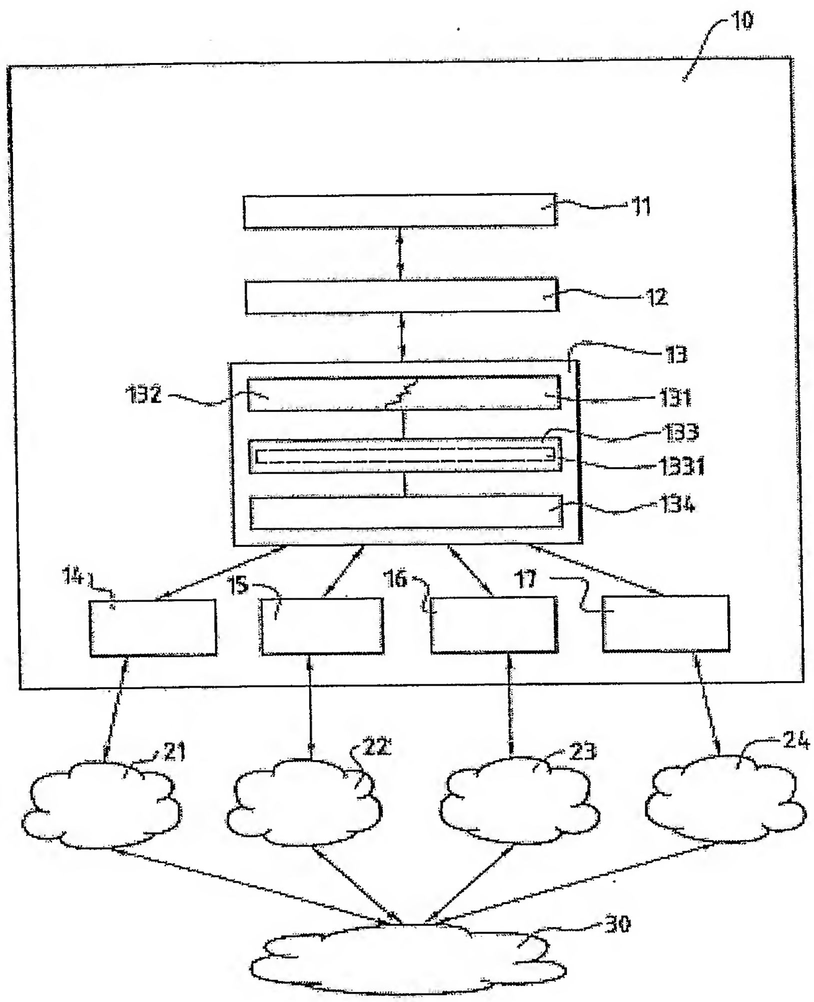


FIG.2

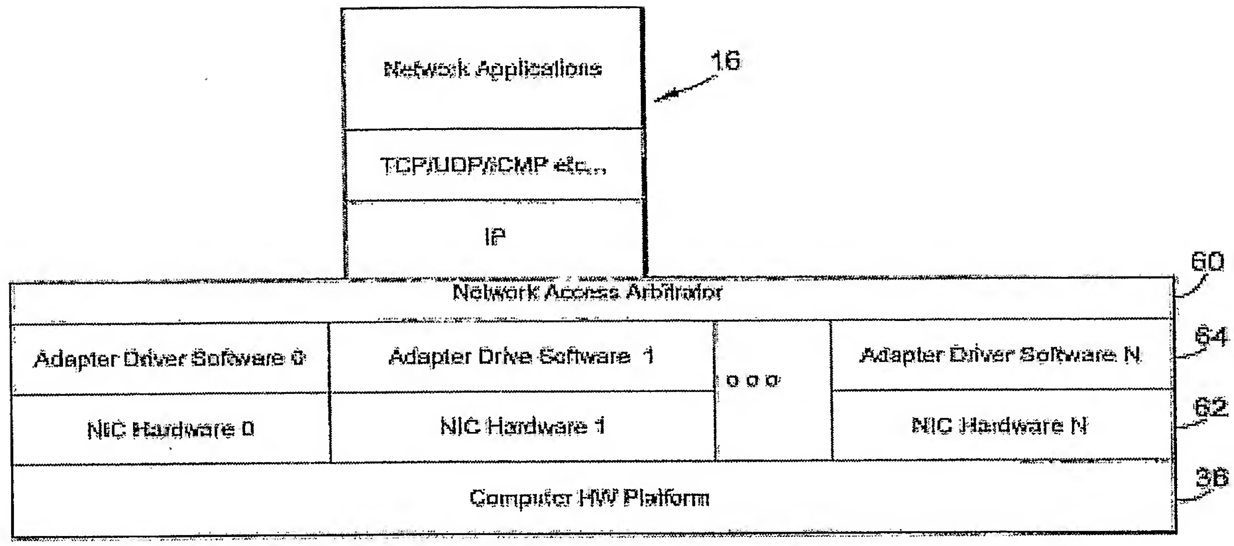

Network Applications
16
TCP/UDP/ICMP etc...
IP
Network Access Arbitrator
60
Adapter Driver Software 0
Adapter Drive Software 1
o o o
Adapter Driver Software N
64
NIC Hardware 0
NIC Hardware 1
NIC Hardware N
62
Computer HW Platform
36

FIG.3

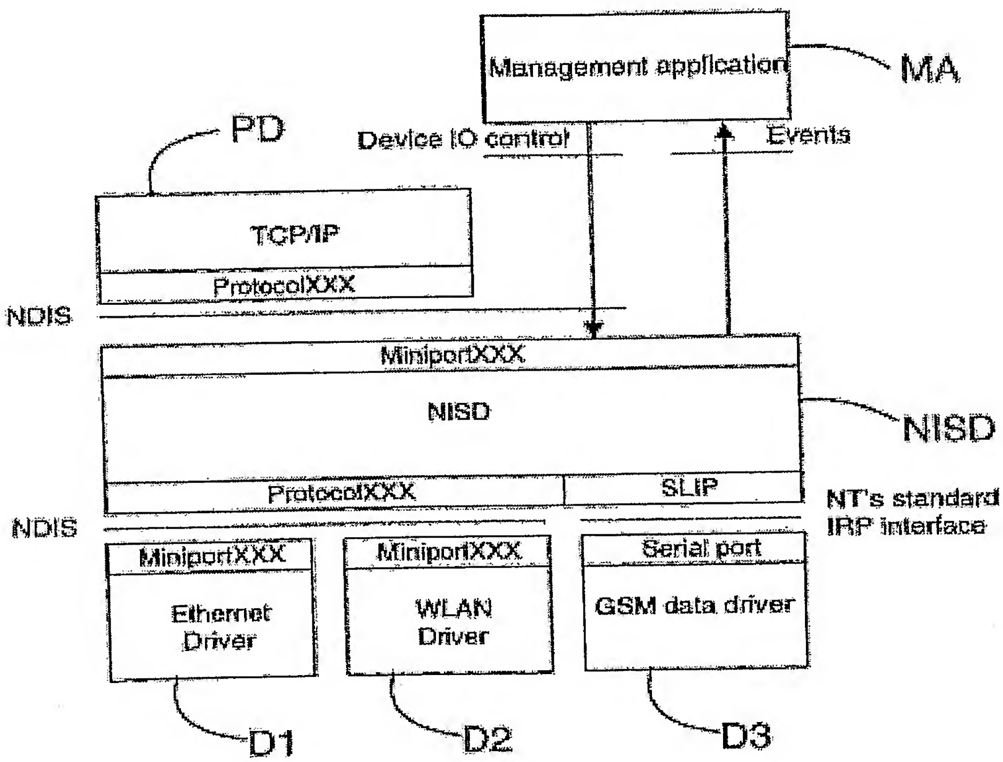
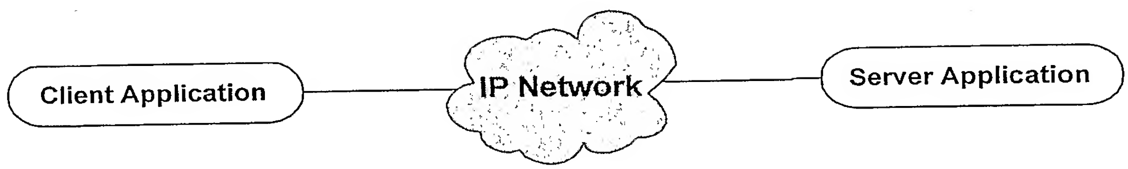

Management application
MA
PD
Device IO control
Events
TCP/IP
ProtocolXXX
NDIS
MiniportXXX
NISD
NISD
ProtocolXXX
SLIP
NT's standard
IRP interface
NDIS
MiniportXXX
Ethernet
Driver
MiniportXXX
WLAN
Driver
Serial port
GSM data driver
D1
D2
D3

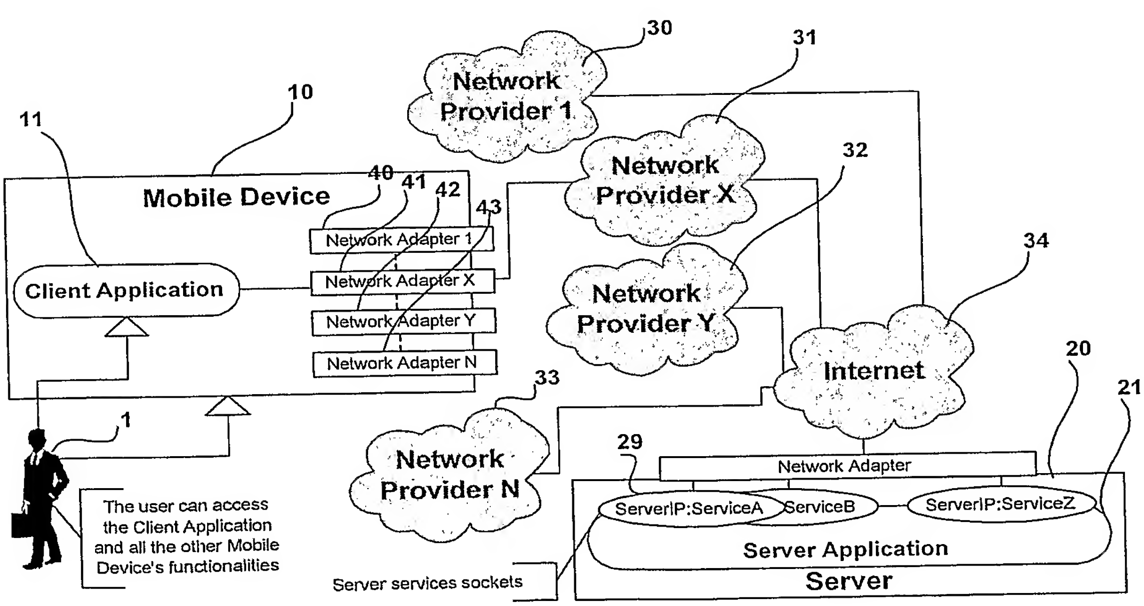
FIG.4

5



Client Application
IP Network
Server Application

FIG.5



30
31
Network Provider 1
10
11
Network Provider X
32
Mobile Device
40
41
42
43
Network Adapter 1
Network Adapter X
Network Adapter Y
Network Adapter N
Client Application
Network Provider Y
34
Internet
20
21
33
1
Network Provider N
29
Network Adapter
ServerIP:ServiceA
ServiceB
ServerIP:ServiceZ
Server Application
Server
The user can access the Client Application and all the other Mobile Device's functionalities
Server services sockets

FIG.6

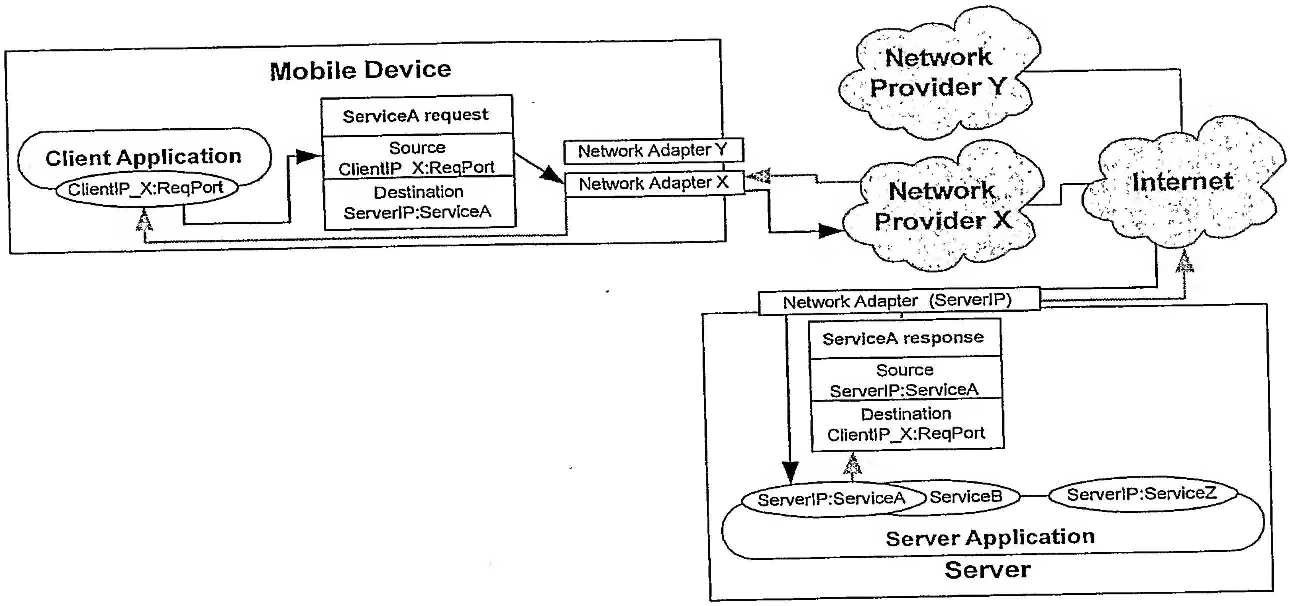

Mobile Device
ServiceA request
Source
ClientIP_X:ReqPort
Destination
ServerIP:ServiceA
Client Application
ClientIP_X:ReqPort
Network Adapter Y
Network Adapter X
Network Provider Y
Network Provider X
Internet
Network Adapter (ServerIP)
ServiceA response
Source
ServerIP:ServiceA
Destination
ClientIP_X:ReqPort
ServerIP:ServiceA
ServiceB
ServerIP:ServiceZ
Server Application
Server

FIG.7

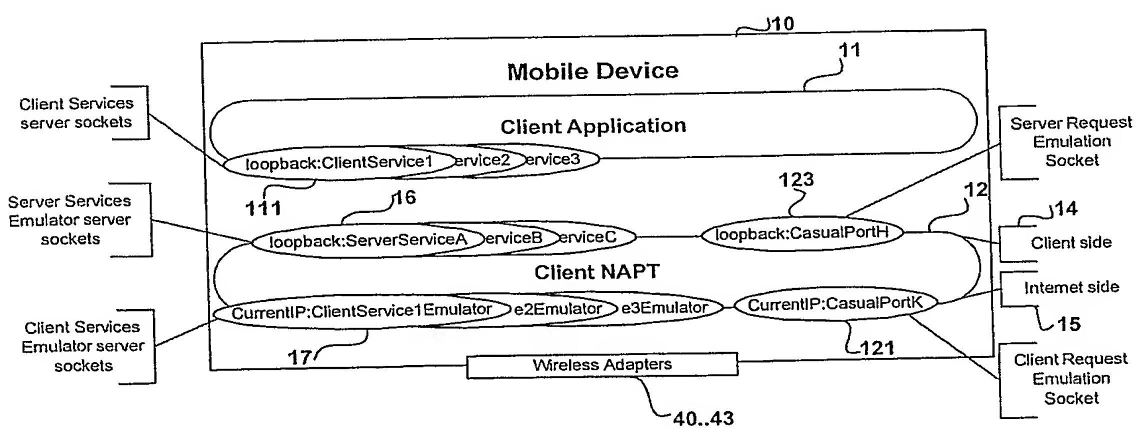

10
11
Mobile Device
Client Services server sockets
Client Application
loopback:ClientService1
ervice2
ervice3
Server Request Emulation Socket
111
16
123
12
14
Server Services Emulator server sockets
loopback:ServerServiceA
erviceB
erviceC
loopback:CasualPortH
Client side
Client NAPT
Internet side
CurrentIP:ClientService1Emulator
e2Emulator
e3Emulator
CurrentIP:CasualPortK
15
Client Services Emulator server sockets
17
121
Wireless Adapters
Client Request Emulation Socket
40..43

FIG.8

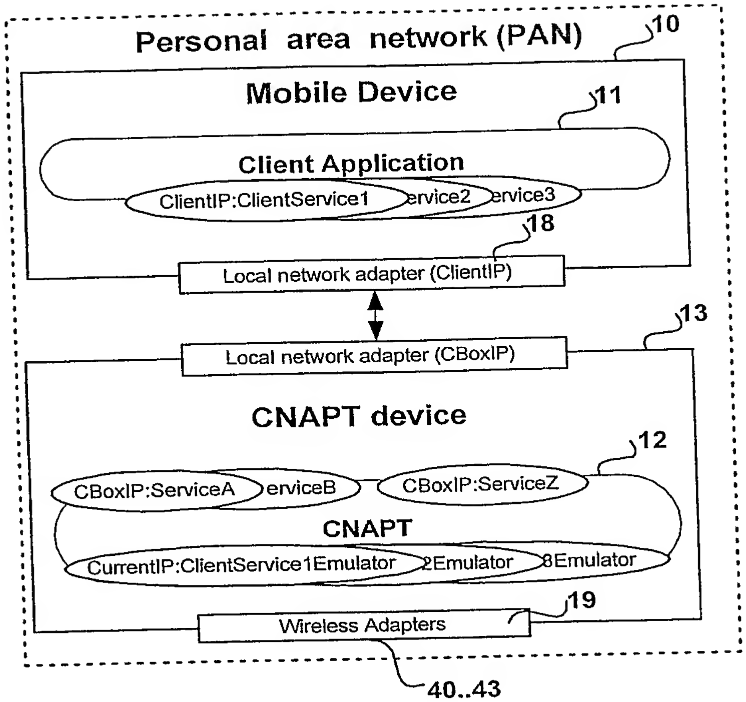

Personal area network (PAN)
10
Mobile Device
11
Client Application
ClientIP:ClientService1
ervice2
ervice3
18
Local network adapter (ClientIP)
13
Local network adapter (CBoxIP)
CNAPT device
12
CBoxIP:ServiceA
erviceB
CBoxIP:ServiceZ
CNAPT
CurrentIP:ClientService1Emulator
2Emulator
3Emulator
19
Wireless Adapters
40..43

FIG.9

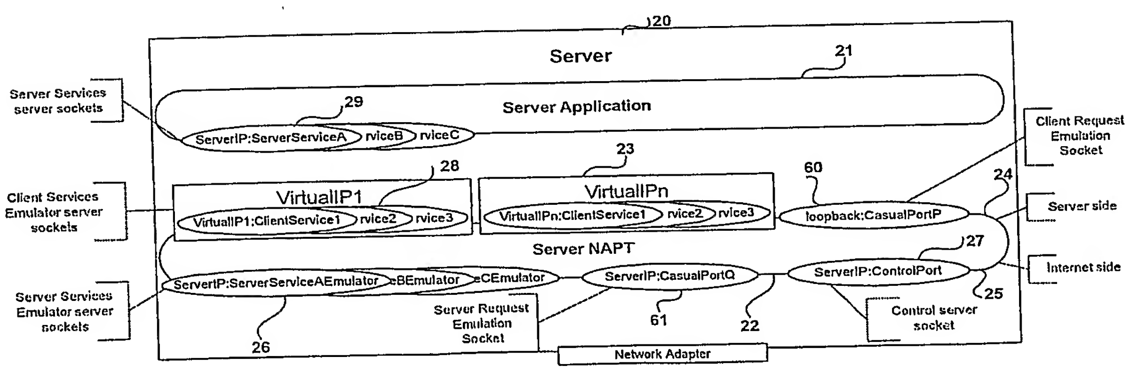

20
Server
21
Server Services server sockets
29
Server Application
ServerIP:ServerServiceA
rviceB
rviceC
Client Request Emulation Socket
23
28
60
24
VirtualIP1
VirtualIP1:ClientService1
rvice2
rvice3
VirtualIPn
VirtualIPn:ClientService1
rvice2
rvice3
loopback:CasualPortP
Client Services Emulator server sockets
Server side
Server NAPT
27
ServerIP:ServerServiceAEmulator
BEmulator
CEmulator
ServerIP:CasualPortQ
ServerIP:ControlPort
Internet side
25
Server Services Emulator server sockets
Server Request Emulation Socket
61
22
Control server socket
26
Network Adapter

FIG.10

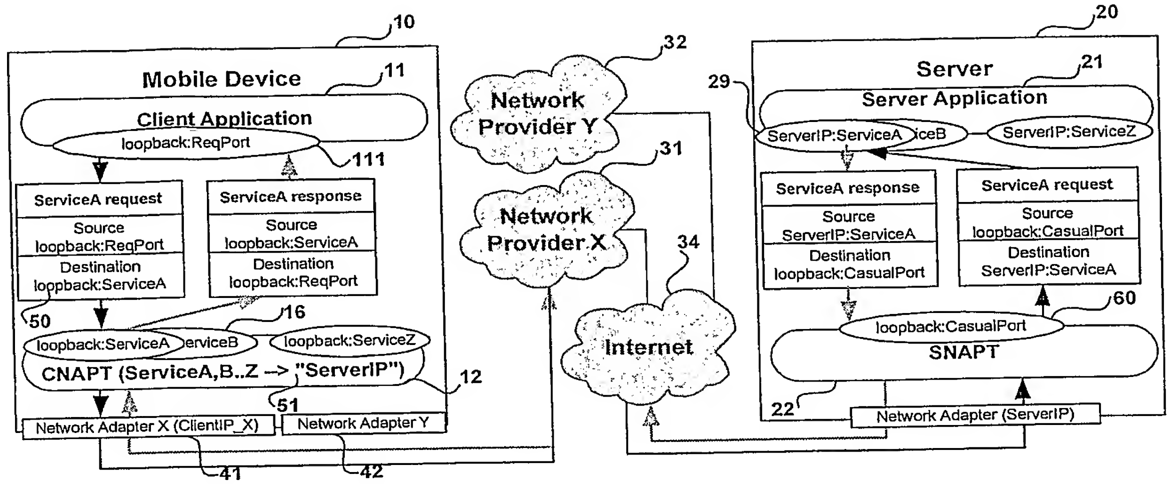

10
Mobile Device
11
Client Application
loopback:ReqPort
111
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPort
50
16
loopback:ServiceA
erviceB
loopback:ServiceZ
CNAPT (ServiceA,B..Z --> "ServerIP")
12
51
Network Adapter X (ClientIP_X)
Network Adapter Y
41
42
32
Network
Provider Y
31
Network
Provider X
34
Internet
20
Server
21
29
Server Application
ServerIP:ServiceA
iceB
ServerIP:ServiceZ
ServiceA response
Source
ServerIP:ServiceA
Destination
loopback:CasualPort
ServiceA request
Source
loopback:CasualPort
Destination
ServerIP:ServiceA
60
loopback:CasualPort
SNAPT
22
Network Adapter (ServerIP)

FIG.11

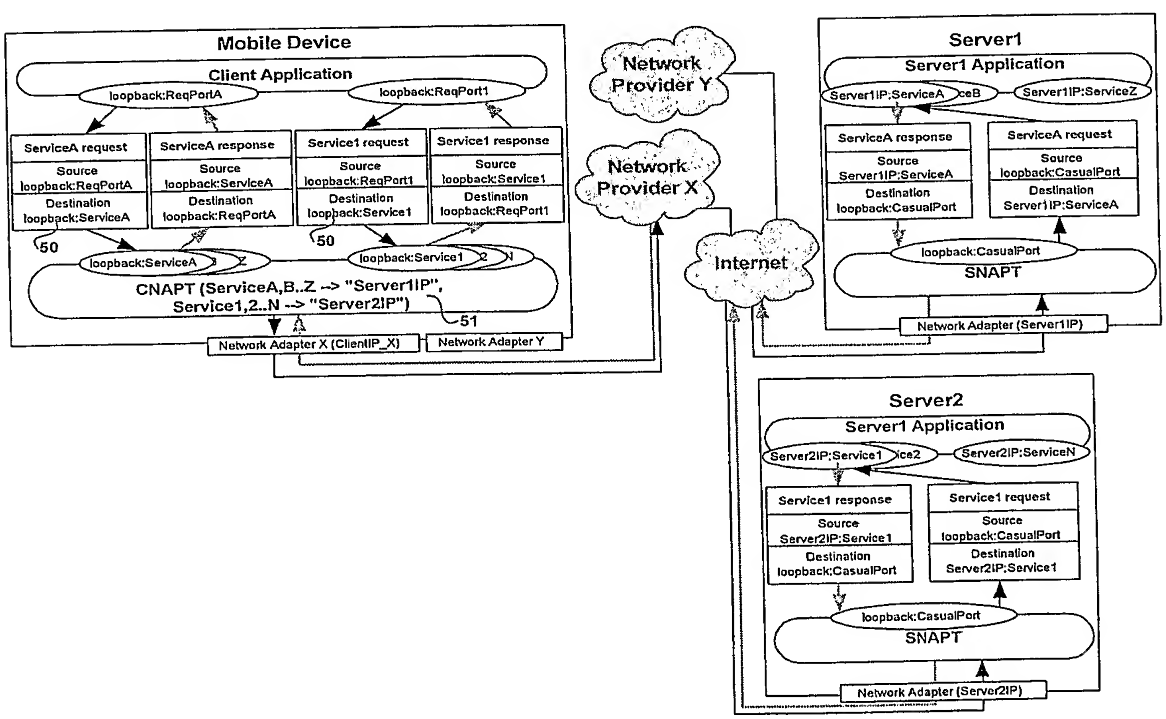

Mobile Device
Client Application
loopback:ReqPortA
loopback:ReqPort1
ServiceA request
Source
loopback:ReqPortA
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPortA
Service1 request
Source
loopback:ReqPort1
Destination
loopback:Service1
Service1 response
Source
loopback:Service1
Destination
loopback:ReqPort1
50
50
loopback:ServiceA
loopback:Service1
CNAPT (ServiceA,B..Z --> "Server1IP",
Service1,2..N --> "Server2IP")
51
Network Adapter X (ClientIP_X)
Network Adapter Y
Network
Provider Y
Network
Provider X
Internet
Server1
Server1 Application
Server1IP:ServiceA
Server1IP:ServiceZ
ServiceA response
Source
Server1IP:ServiceA
Destination
loopback:CasualPort
ServiceA request
Source
loopback:CasualPort
Destination
Server1IP:ServiceA
loopback:CasualPort
SNAPT
Network Adapter (Server1IP)
Server2
Server1 Application
Server2IP:Service1
ice2
Server2IP:ServiceN
Service1 response
Source
Server2IP:Service1
Destination
loopback:CasualPort
Service1 request
Source
loopback:CasualPort
Destination
Server2IP:Service1
loopback:CasualPort
SNAPT
Network Adapter (Server2IP)

FIG.12

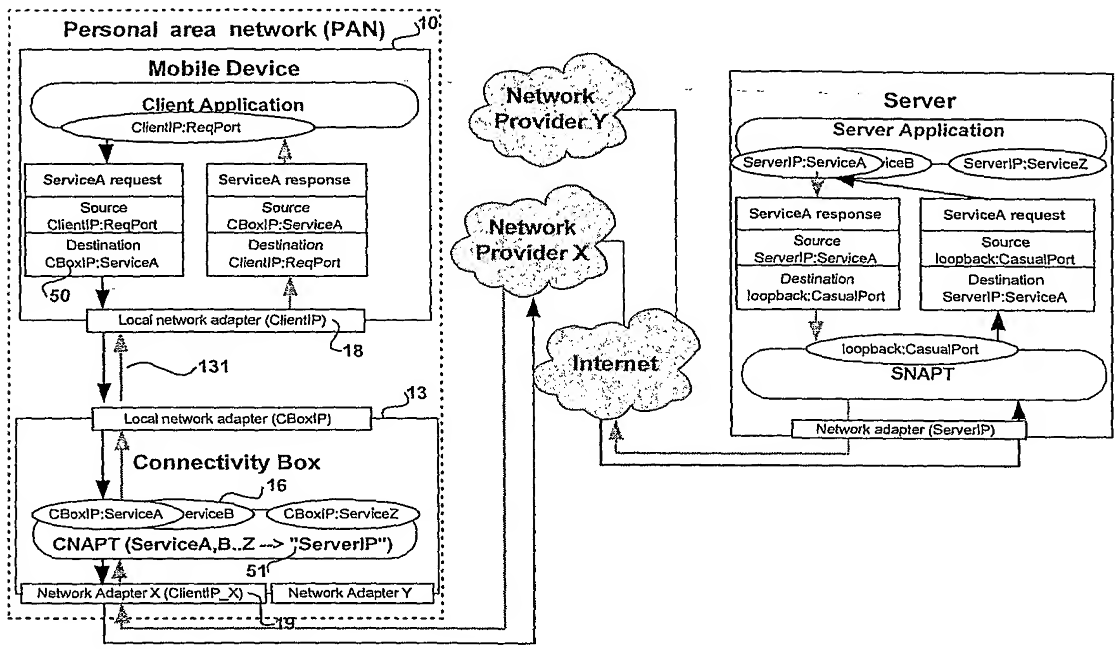
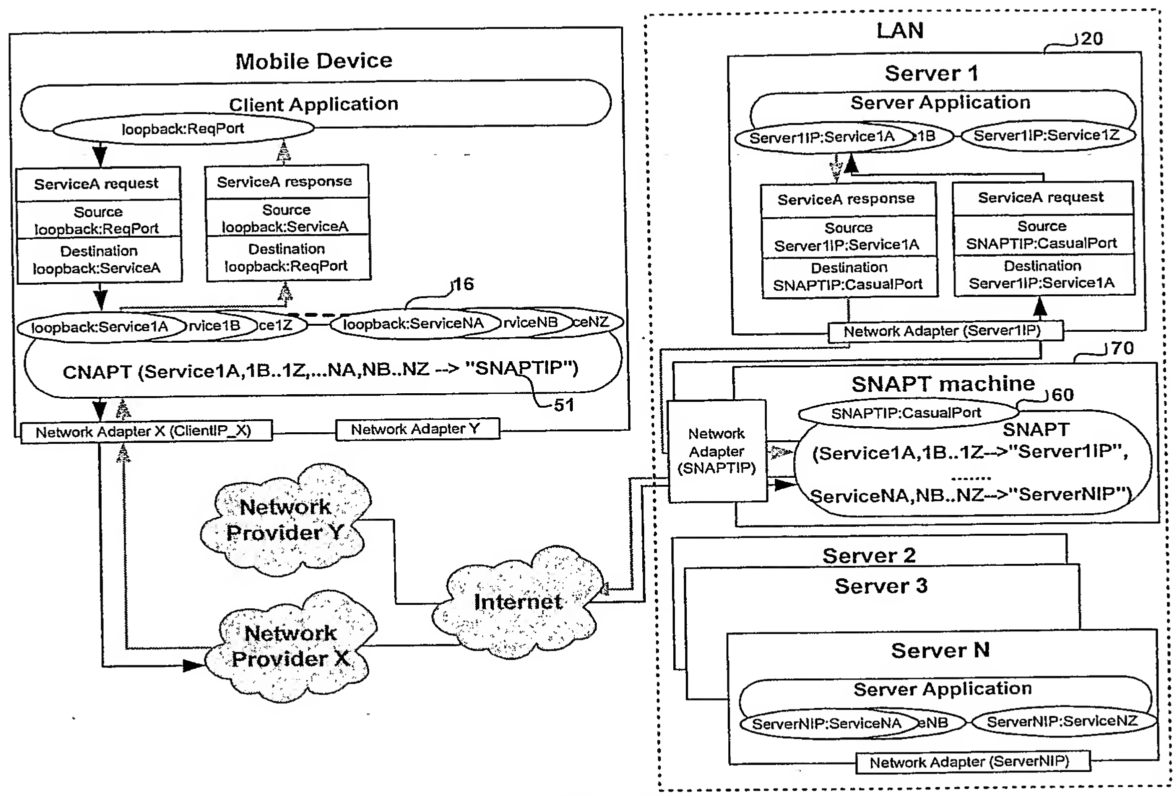
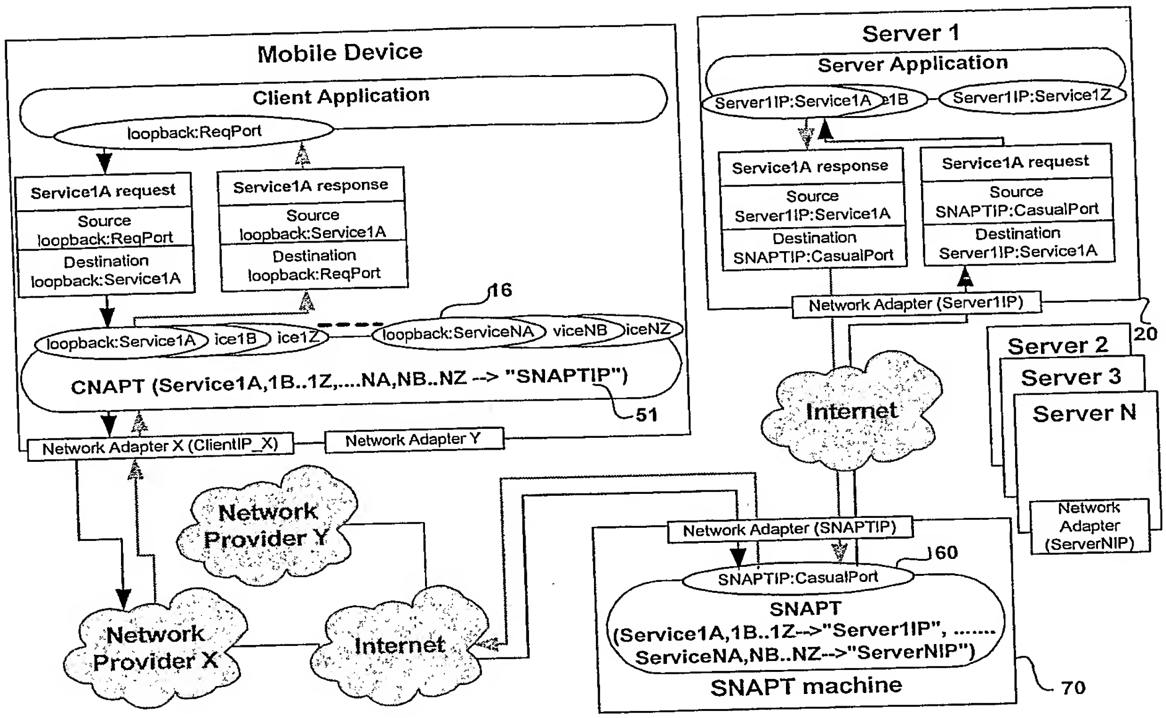

Personal area network (PAN)
10
Mobile Device
Client Application
ClientIP:ReqPort
ServiceA request
Source
ClientIP:ReqPort
Destination
CBoxIP:ServiceA
ServiceA response
Source
CBoxIP:ServiceA
Destination
ClientIP:ReqPort
50
Local network adapter (ClientIP)
18
131
13
Local network adapter (CBoxIP)
Connectivity Box
16
CBoxIP:ServiceA
erviceB
CBoxIP:ServiceZ
CNAPT (ServiceA,B..Z --> "ServerIP")
51
Network Adapter X (ClientIP_X)
Network Adapter Y
19
Network Provider Y
Network Provider X
Internet
Server
Server Application
ServerIP:ServiceA
iceB
ServerIP:ServiceZ
ServiceA response
Source
ServerIP:ServiceA
Destination
loopback:CasualPort
ServiceA request
Source
loopback:CasualPort
Destination
ServerIP:ServiceA
loopback:CasualPort
SNAPT
Network adapter (ServerIP)

FIG.13



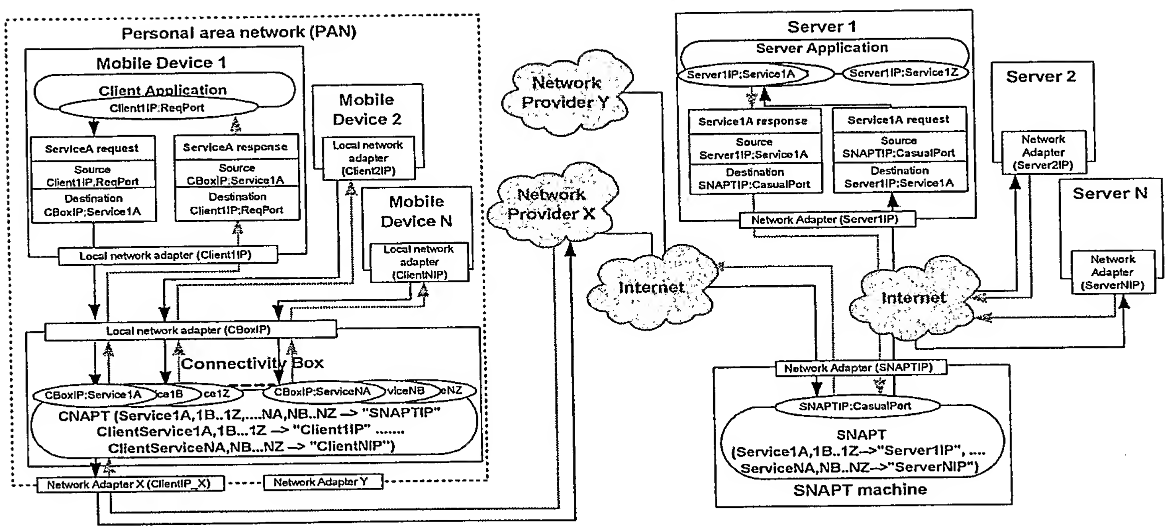
Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPort
16
loopback:Service1A
rvice1B
ce1Z
loopback:ServiceNA
rviceNB
ceNZ
CNAPT (Service1A,1B..1Z,...NA,NB..NZ --> "SNAPTIP")
51
Network Adapter X (ClientIP_X)
Network Adapter Y
Network Provider Y
Network Provider X
Internet
LAN
20
Server 1
Server Application
Server1IP:Service1A
1B
Server1IP:Service1Z
ServiceA response
Source
Server1IP:Service1A
Destination
SNAPTIP:CasualPort
ServiceA request
Source
SNAPTIP:CasualPort
Destination
Server1IP:Service1A
Network Adapter (Server1IP)
70
SNAPT machine
60
SNAPTIP:CasualPort
Network Adapter (SNAPTIP)
SNAPT
(Service1A,1B..1Z-->"Server1IP",
....
ServiceNA,NB..NZ-->"ServerNIP")
Server 2
Server 3
Server N
Server Application
ServerNIP:ServiceNA
eNB
ServerNIP:ServiceNZ
Network Adapter (ServerNIP)

FIG.14



Mobile Device
Client Application
loopback:ReqPort
Service1A request
Source
loopback:ReqPort
Destination
loopback:Service1A
Service1A response
Source
loopback:Service1A
Destination
loopback:ReqPort
16
loopback:Service1A
ice1B
ice1Z
loopback:ServiceNA
viceNB
iceNZ
CNAPT (Service1A,1B..1Z,....NA,NB..NZ --> "SNAPTIP")
51
Network Adapter X (ClientIP_X)
Network Adapter Y
Network Provider Y
Network Provider X
Internet
Server 1
Server Application
Server1IP:Service1A
1B
Server1IP:Service1Z
Service1A response
Source
Server1IP:Service1A
Destination
SNAPTIP:CasualPort
Service1A request
Source
SNAPTIP:CasualPort
Destination
Server1IP:Service1A
Network Adapter (Server1IP)
20
Server 2
Server 3
Server N
Network Adapter (ServerNIP)
Internet
Network Adapter (SNAPTIP)
60
SNAPTIP:CasualPort
SNAPT
(Service1A,1B..1Z-->"Server1IP", ...
ServiceNA,NB..NZ-->"ServerNIP")
SNAPT machine
70

FIG.15



Personal area network (PAN)
Mobile Device 1
Client Application
Client1IP:ReqPort
ServiceA request
Source
Client1IP,ReqPort
Destination
CBoxIP:Service1A
ServiceA response
Source
CBoxIP:Service1A
Destination
Client1IP:ReqPort
Local network adapter (Client1IP)
Mobile Device 2
Local network adapter (Client2IP)
Mobile Device N
Local network adapter (ClientNIP)
Local network adapter (CBoxIP)
Connectivity Box
CBoxIP:Service1A
CBoxIP:ServiceNA
CNAPT (Service1A,1B..1Z,....NA,NB..NZ -> "SNAPTIP"
ClientService1A,1B...1Z -> "Client1IP" ....
ClientServiceNA,NB...NZ -> "ClientNIP")
Network Adapter X (ClientIP_X)
Network Adapter Y
Network Provider Y
Network Provider X
Internet
Server 1
Server Application
Server1IP:Service1A
Server1IP:Service1Z
Service1A response
Source
Server1IP:Service1A
Destination
SNAPTIP:CasualPort
Service1A request
Source
SNAPTIP:CasualPort
Destination
Server1IP:Service1A
Network Adapter (Server1IP)
Server 2
Network Adapter (Server2IP)
Server N
Network Adapter (ServerNIP)
Internet
Network Adapter (SNAPTIP)
SNAPTIP:CasualPort
SNAPT
(Service1A,1B..1Z->"Server1IP", ...
ServiceNA,NB..NZ->"ServerNIP")
SNAPT machine

FIG.16

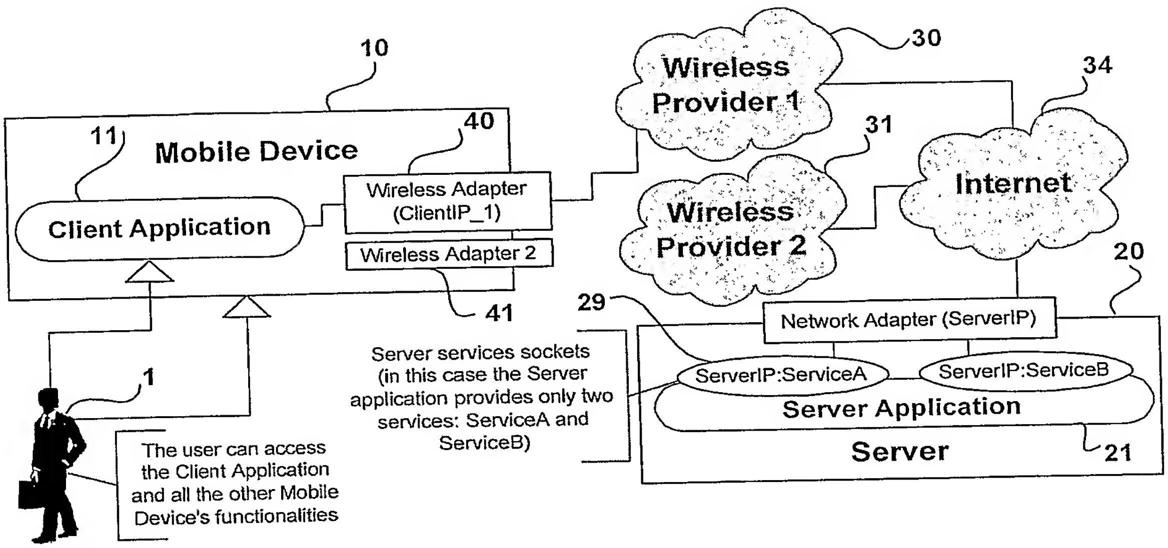

10
11
Mobile Device
40
Client Application
Wireless Adapter (ClientIP_1)
Wireless Adapter 2
41
1
The user can access the Client Application and all the other Mobile Device's functionalities
Wireless Provider 1
30
Wireless Provider 2
31
Internet
34
20
29
Server services sockets (in this case the Server application provides only two services: ServiceA and ServiceB)
Network Adapter (ServerIP)
ServerIP:ServiceA
ServerIP:ServiceB
Server Application
Server
21

FIG.17

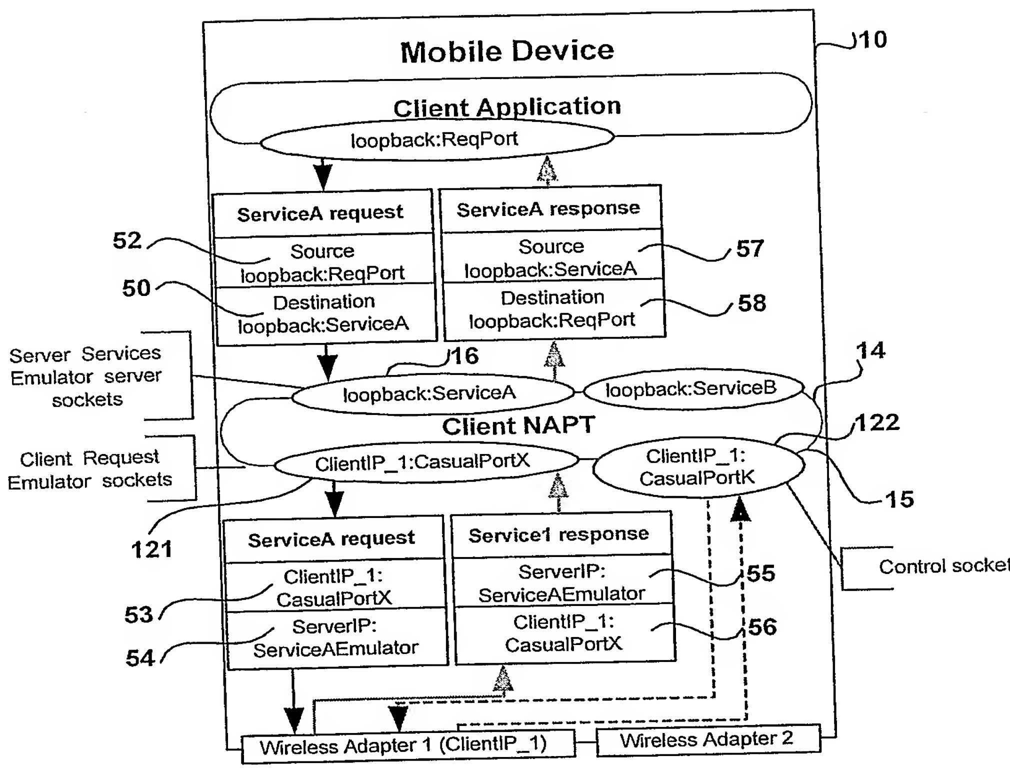

Mobile Device
10
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPort
52
50
57
58
Server Services Emulator server sockets
16
14
loopback:ServiceA
loopback:ServiceB
Client NAPT
122
Client Request Emulator sockets
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
15
121
ServiceA request
ClientIP_1:
CasualPortX
ServerIP:
ServiceAEmulator
Service1 response
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
55
56
53
54
Control socket
Wireless Adapter 1 (ClientIP_1)
Wireless Adapter 2

FIG.18

5

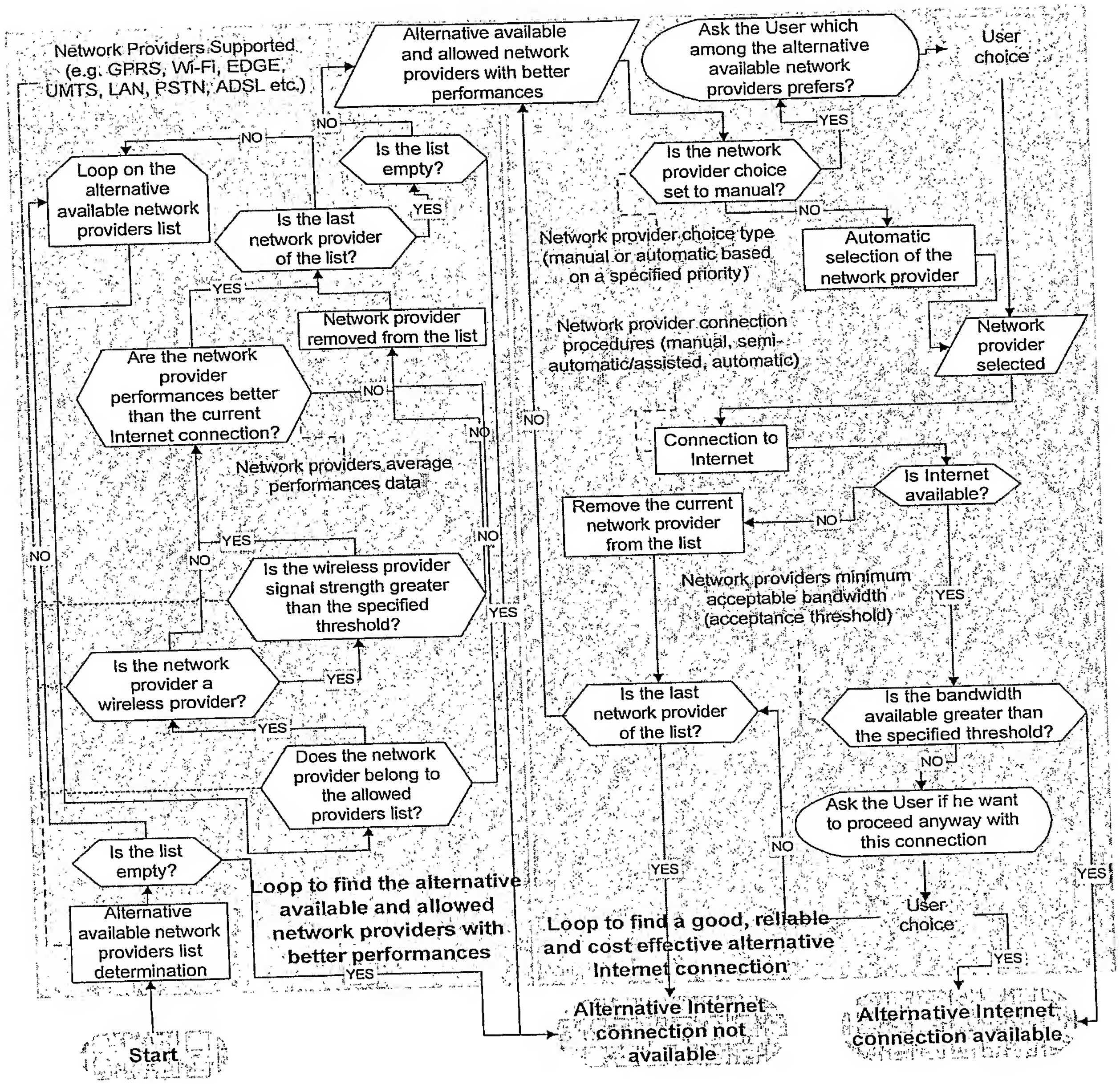

Network Providers Supported (e.g. GPRS, Wi-Fi, EDGE, UMTS, LAN, PSTN, ADSL etc.)
Alternative available and allowed network providers with better performances
Ask the User which among the alternative available network providers prefers?
User choice
NO
YES
Is the list empty?
Loop on the alternative available network providers list
Is the last network provider of the list?
Is the network provider choice set to manual?
Network provider choice type (manual or automatic based on a specified priority)
Automatic selection of the network provider
Network provider removed from the list
Network provider connection procedures (manual, semi-automatic/assisted, automatic)
Network provider selected
Are the network provider performances better than the current Internet connection?
Connection to Internet
Is Internet available?
Network providers average performances data
Remove the current network provider from the list
Is the wireless provider signal strength greater than the specified threshold?
Network providers minimum acceptable bandwidth (acceptance threshold)
Is the network provider a wireless provider?
Is the last network provider of the list?
Is the bandwidth available greater than the specified threshold?
Does the network provider belong to the allowed providers list?
Ask the User if he want to proceed anyway with this connection
Is the list empty?
Loop to find the alternative available and allowed network providers with better performances
Alternative available network providers list determination
Loop to find a good, reliable and cost effective alternative Internet connection
User choice
Alternative Internet connection not available
Alternative Internet connection available
Start

FIG.19

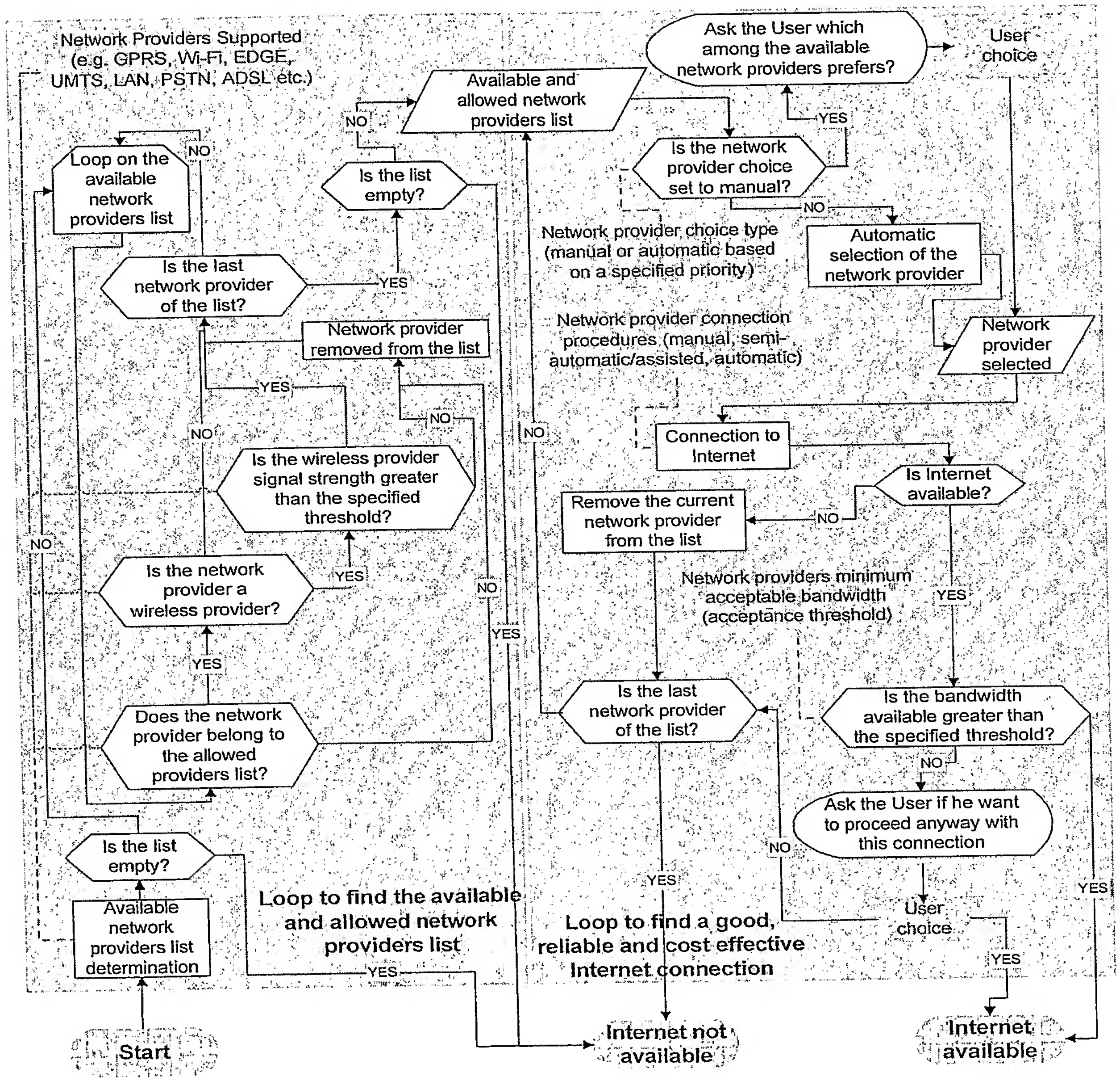

Network Providers Supported (e.g. GPRS, Wi-Fi, EDGE, UMTS, LAN, PSTN, ADSL etc.)
Available and allowed network providers list
Ask the User which among the available network providers prefers?
User choice
Loop on the available network providers list
Is the list empty?
Is the network provider choice set to manual?
Is the last network provider of the list?
Network provider choice type (manual or automatic based on a specified priority)
Automatic selection of the network provider
Network provider removed from the list
Network provider connection procedures (manual, semi-automatic/assisted, automatic)
Network provider selected
Is the wireless provider signal strength greater than the specified threshold?
Connection to Internet
Is Internet available?
Remove the current network provider from the list
Is the network provider a wireless provider?
Network providers minimum acceptable bandwidth (acceptance threshold)
Does the network provider belong to the allowed providers list?
Is the last network provider of the list?
Is the bandwidth available greater than the specified threshold?
Ask the User if he want to proceed anyway with this connection
Is the list empty?
Available network providers list determination
Loop to find the available and allowed network providers list
Loop to find a good, reliable and cost effective Internet connection
User choice
Start
Internet not available
Internet available
YES
NO

FIG.20

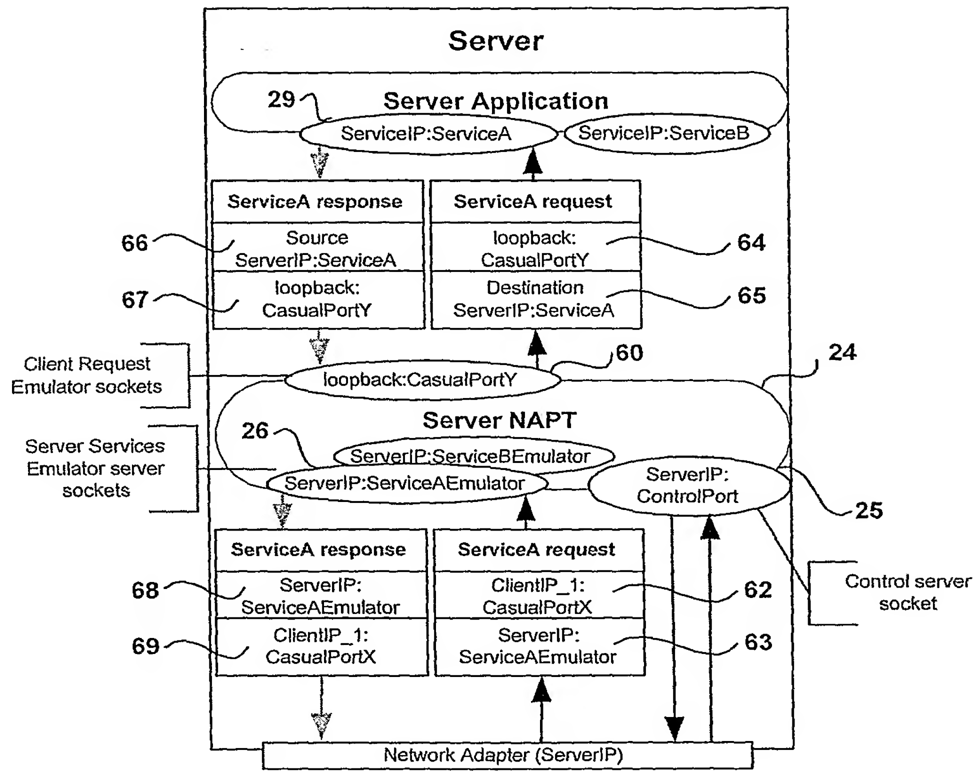
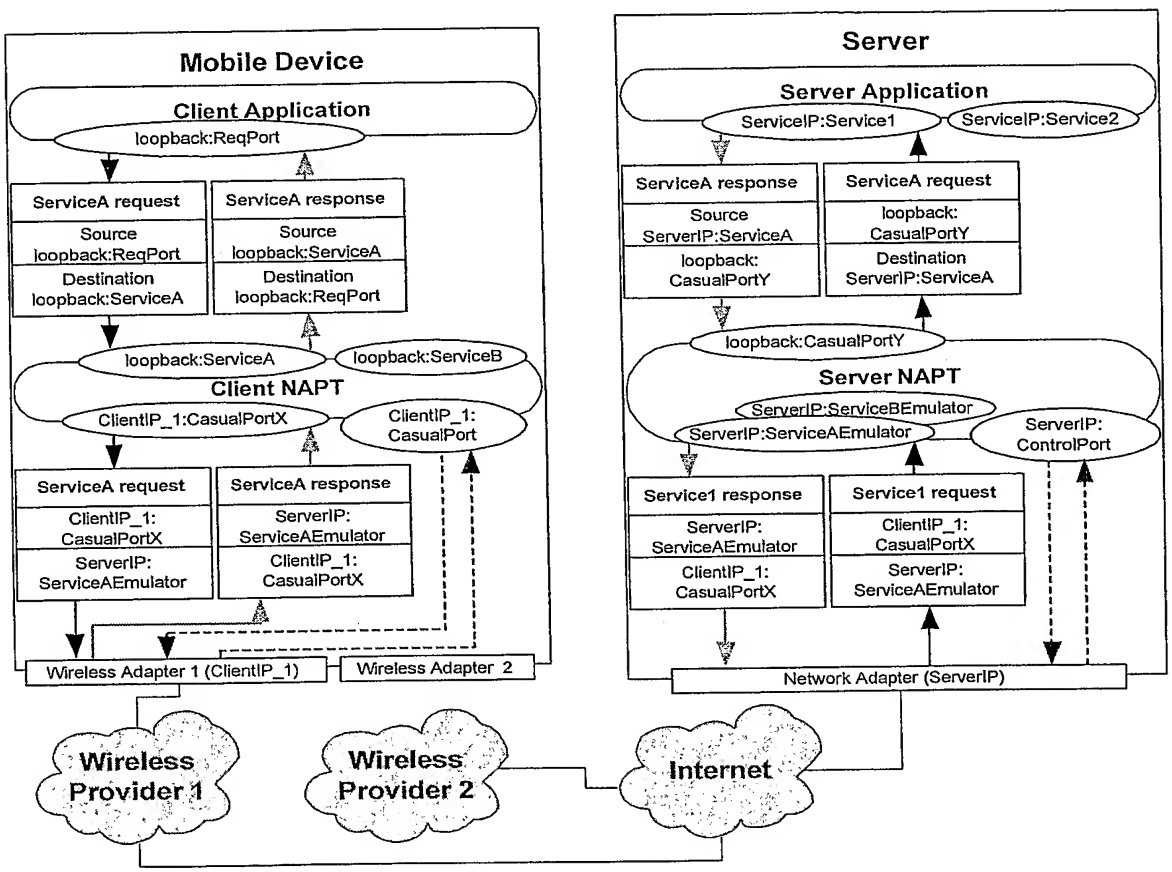
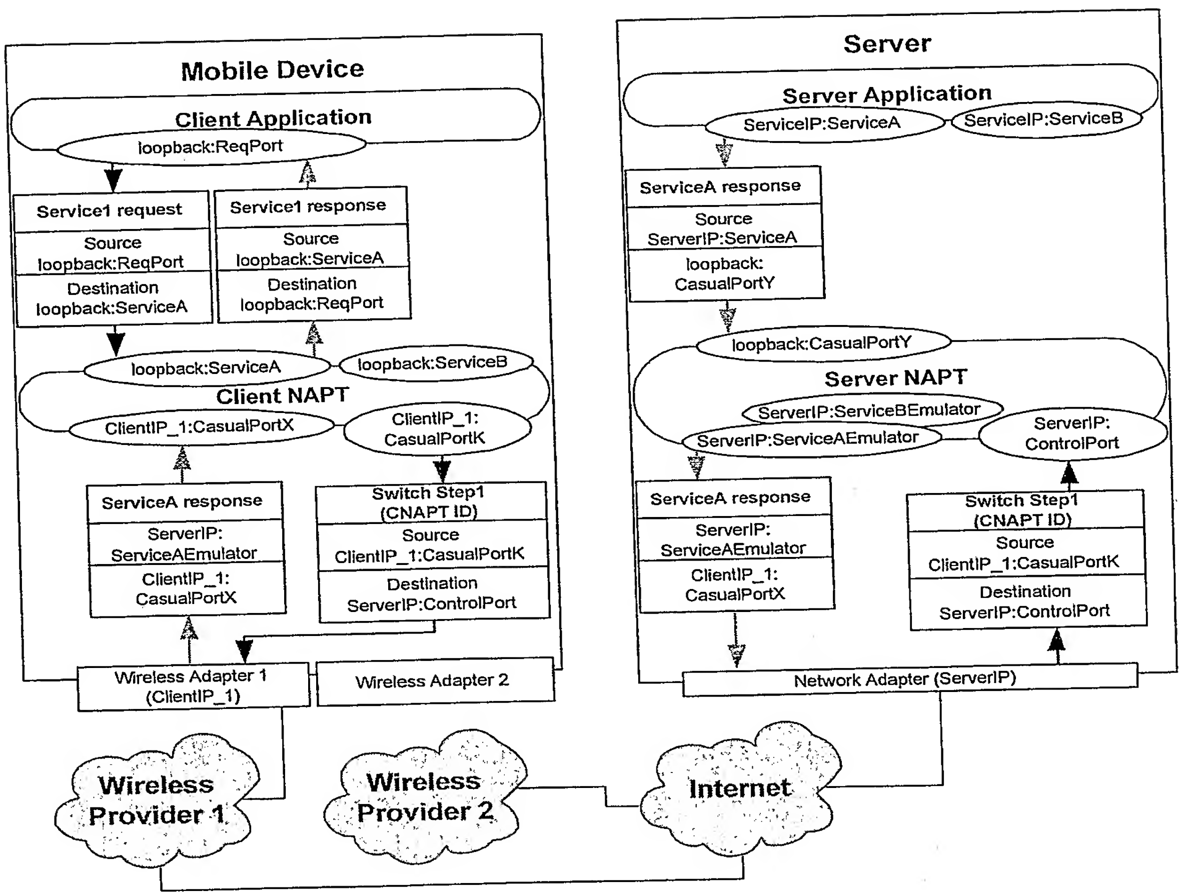

Server
29
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source ServerIP:ServiceA
loopback: CasualPortY
ServiceA request
loopback: CasualPortY
Destination ServerIP:ServiceA
66
67
64
65
60
24
Client Request Emulator sockets
loopback:CasualPortY
Server NAPT
26
Server Services Emulator server sockets
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP: ControlPort
25
ServiceA response
ServerIP: ServiceAEmulator
ClientIP_1: CasualPortX
ServiceA request
ClientIP_1: CasualPortX
ServerIP: ServiceAEmulator
68
69
62
63
Control server socket
Network Adapter (ServerIP)

FIG.21



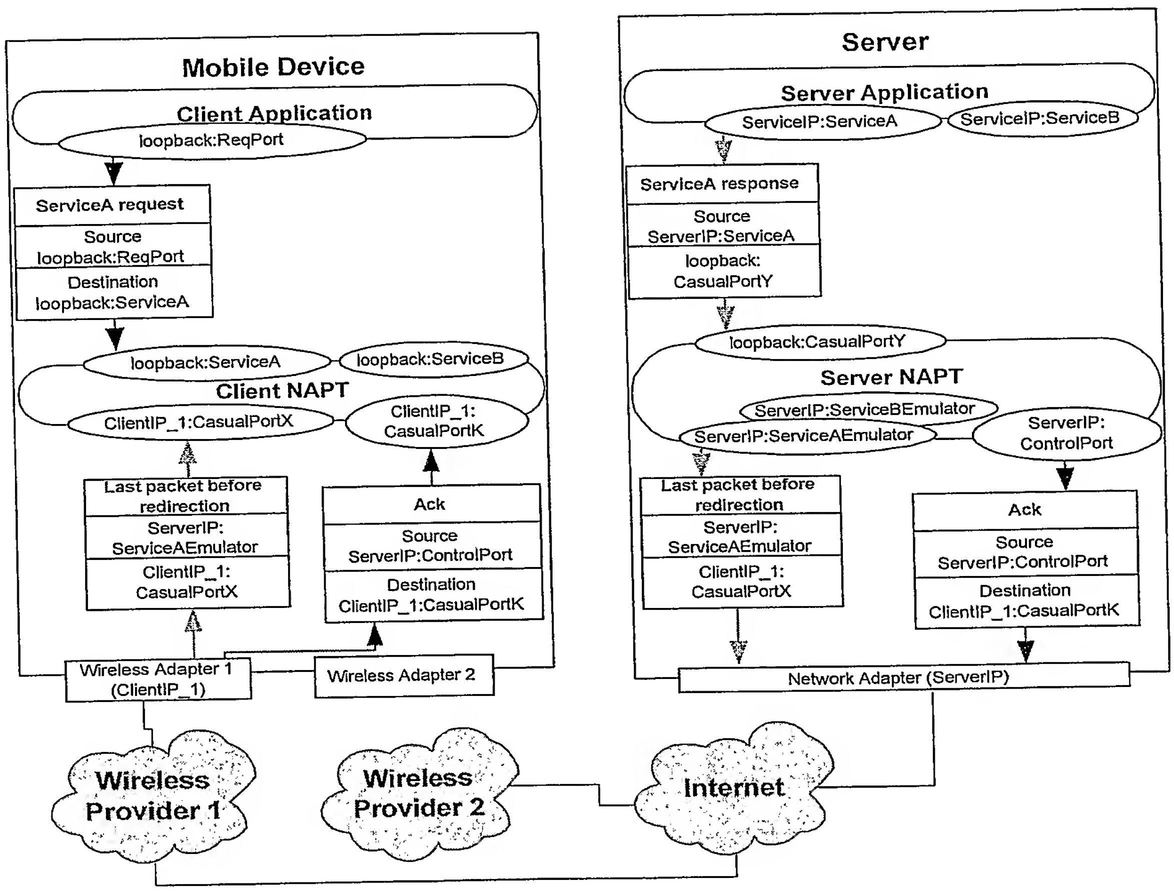
Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPort
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPort
ServiceA request
ClientIP_1:
CasualPortX
ServerIP:
ServiceAEmulator
ServiceA response
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Wireless Adapter 1 (ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:Service1
ServiceIP:Service2
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
ServiceA request
loopback:
CasualPortY
Destination
ServerIP:ServiceA
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Service1 response
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Service1 request
ClientIP_1:
CasualPortX
ServerIP:
ServiceAEmulator
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.22



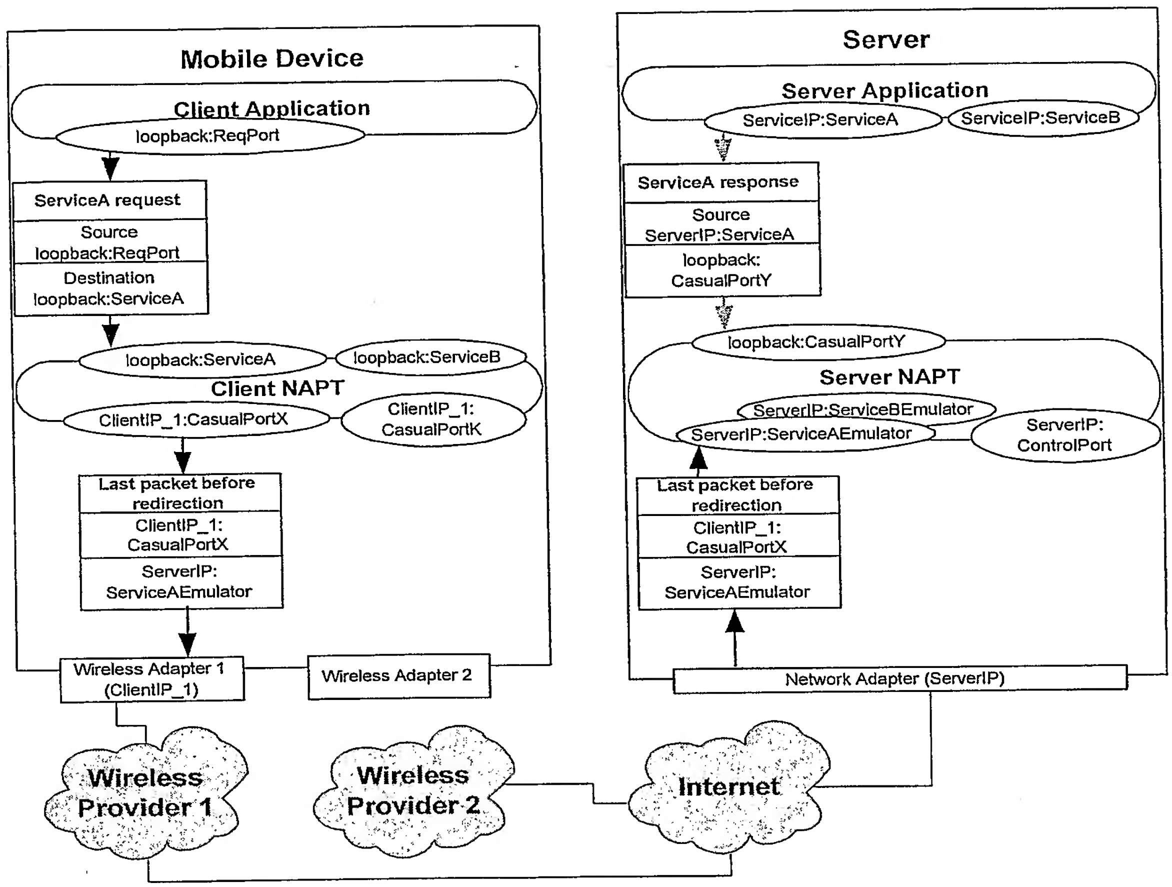
Mobile Device
Client Application
loopback:ReqPort
Service1 request
Source
loopback:ReqPort
Destination
loopback:ServiceA
Service1 response
Source
loopback:ServiceA
Destination
loopback:ReqPort
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
ServiceA response
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Switch Step1
(CNAPT ID)
Source
ClientIP_1:CasualPortK
Destination
ServerIP:ControlPort
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
ServiceA response
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Switch Step1
(CNAPT ID)
Source
ClientIP_1:CasualPortK
Destination
ServerIP:ControlPort
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.23



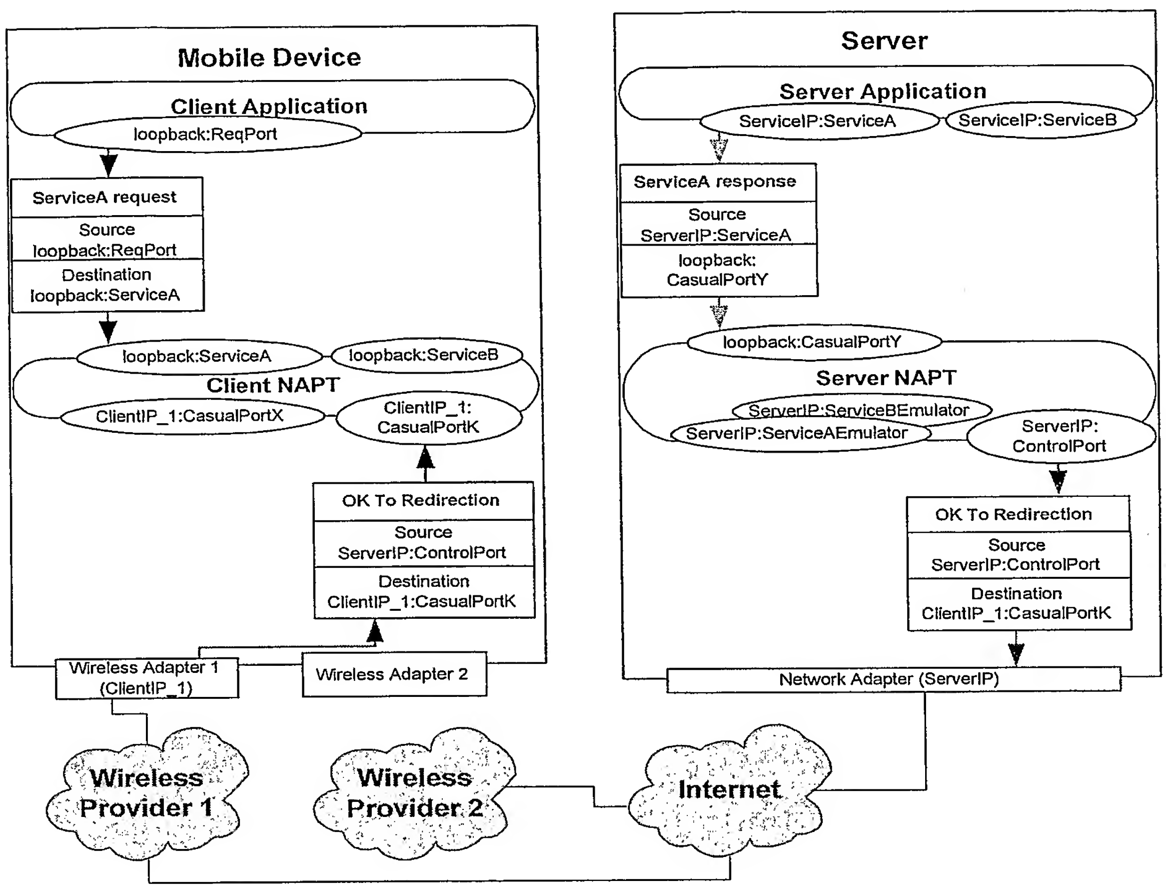
Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
Last packet before redirection
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Ack
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Last packet before redirection
ServerIP:
ServiceAEmulator
ClientIP_1:
CasualPortX
Ack
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Network Adapter (ServerIP)
Wireless Provider 1
Wireless Provider 2
Internet

FIG.24



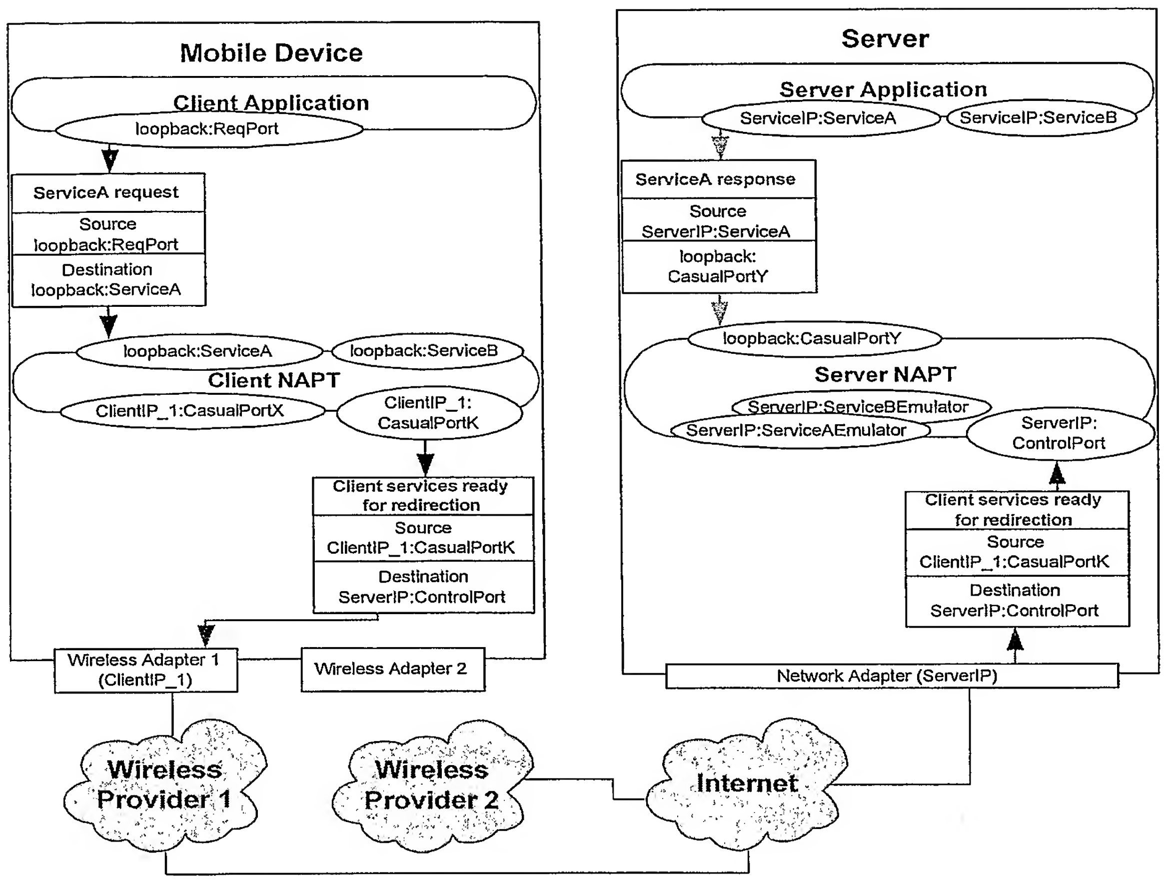
Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
Last packet before redirection
ClientIP_1:
CasualPortX
ServerIP:
ServiceAEmulator
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Last packet before redirection
ClientIP_1:
CasualPortX
ServerIP:
ServiceAEmulator
Network Adapter (ServerIP)
Wireless Provider 1
Wireless Provider 2
Internet

FIG.25



Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
OK To Redirection
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
OK To Redirection
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.26



Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
Client services ready for redirection
Source
ClientIP_1:CasualPortK
Destination
ServerIP:ControlPort
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Client services ready for redirection
Source
ClientIP_1:CasualPortK
Destination
ServerIP:ControlPort
Network Adapter (ServerIP)
Wireless Provider 1
Wireless Provider 2
Internet

FIG.27

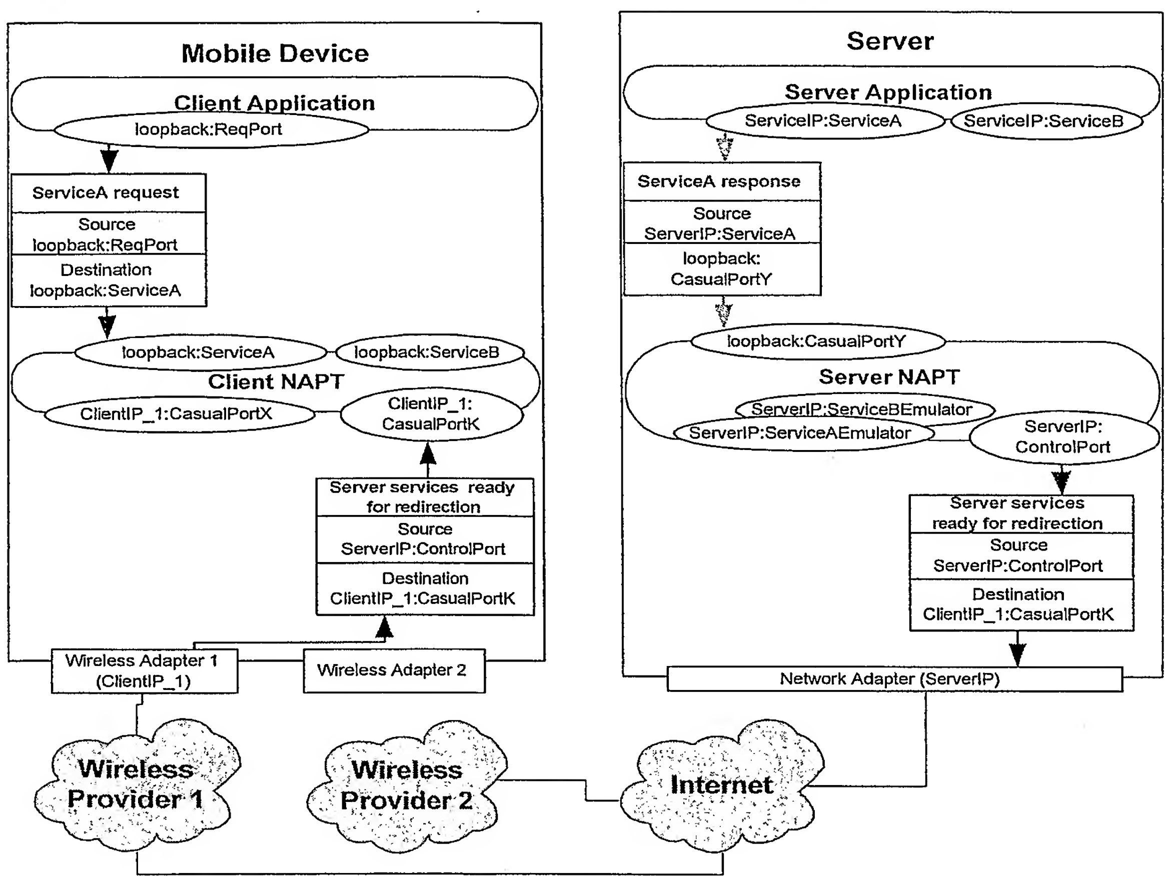
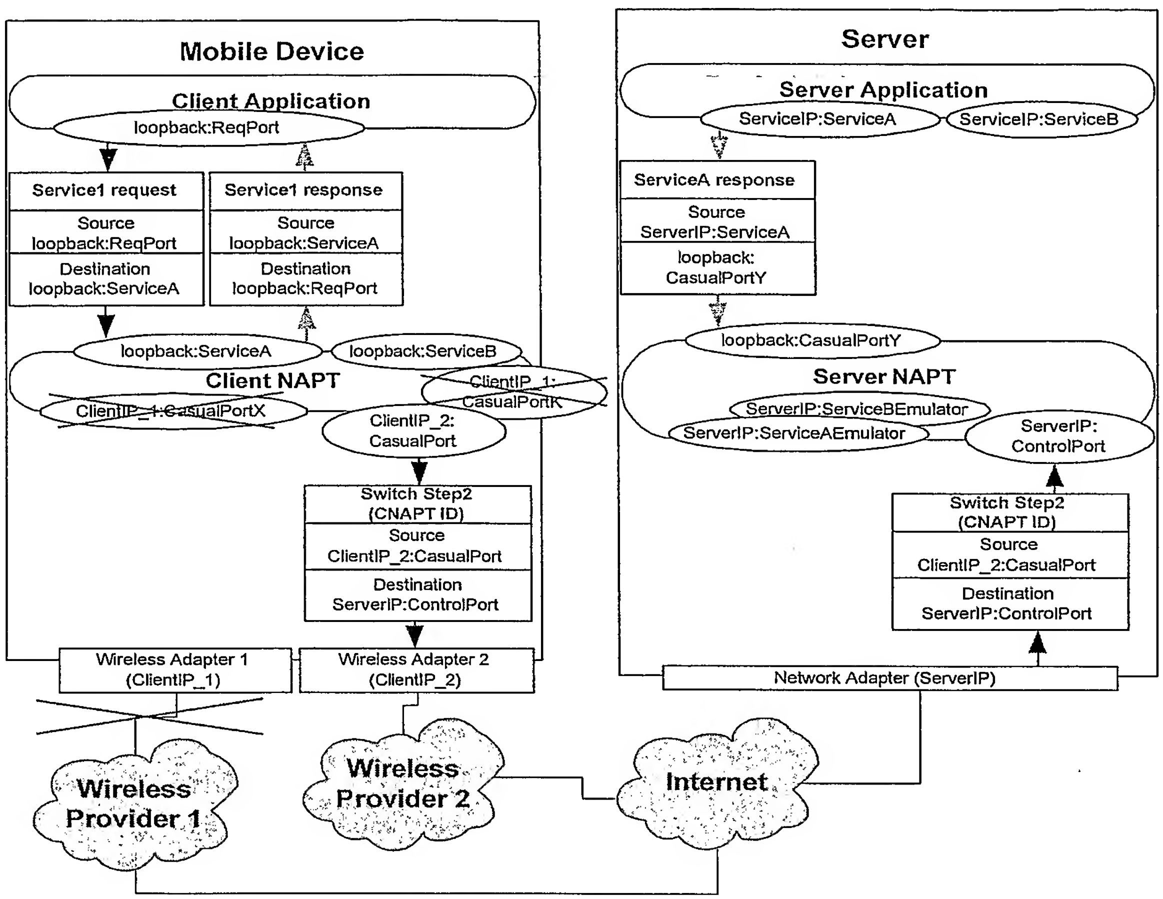
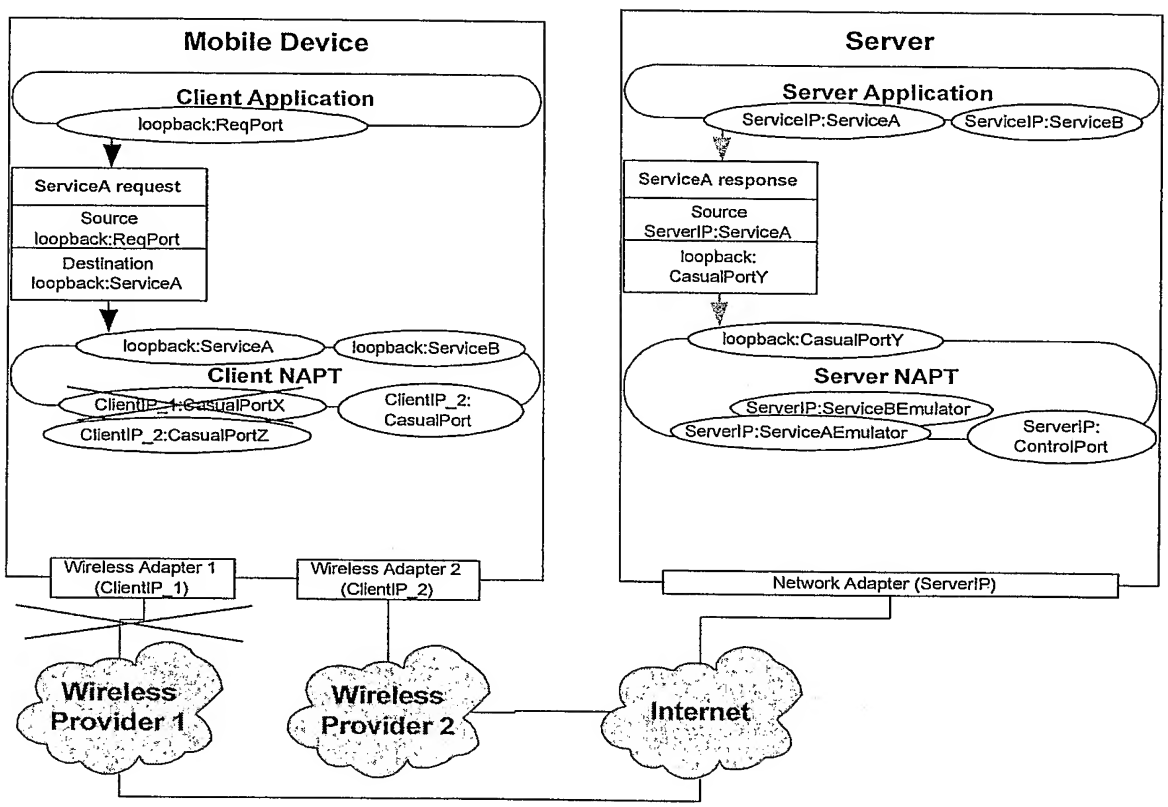

Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
Server services ready
for redirection
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Server services
ready for redirection
Source
ServerIP:ControlPort
Destination
ClientIP_1:CasualPortK
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.28



Mobile Device
Client Application
loopback:ReqPort
Service1 request
Source
loopback:ReqPort
Destination
loopback:ServiceA
Service1 response
Source
loopback:ServiceA
Destination
loopback:ReqPort
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_1:
CasualPortK
ClientIP_2:
CasualPort
Switch Step2
(CNAPT ID)
Source
ClientIP_2:CasualPort
Destination
ServerIP:ControlPort
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
(ClientIP_2)
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Switch Step2
(CNAPT ID)
Source
ClientIP_2:CasualPort
Destination
ServerIP:ControlPort
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.29



Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_1:CasualPortX
ClientIP_2:
CasualPort
ClientIP_2:CasualPortZ
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
(ClientIP_2)
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.30

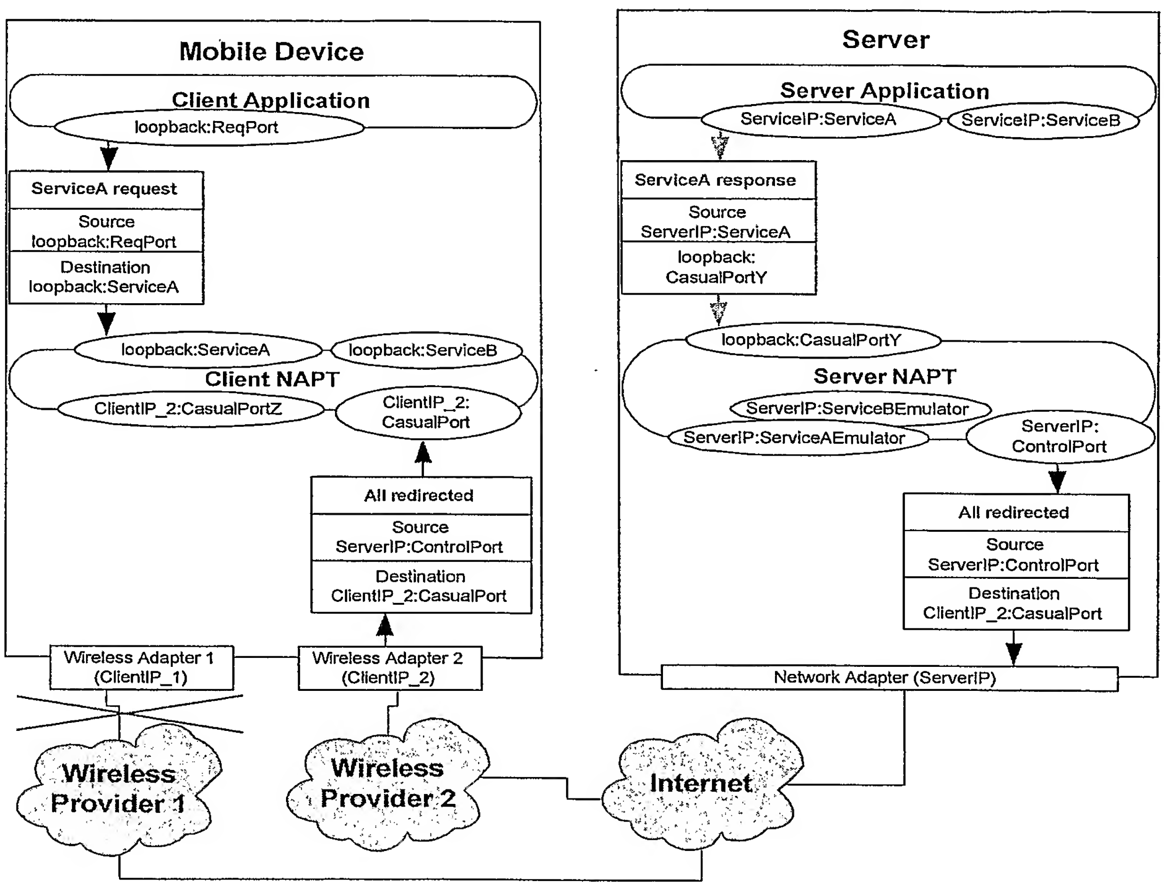

Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_2:CasualPortZ
ClientIP_2:
CasualPort
All redirected
Source
ServerIP:ControlPort
Destination
ClientIP_2:CasualPort
Wireless Adapter 1
(ClientIP_1)
Wireless Adapter 2
(ClientIP_2)
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
All redirected
Source
ServerIP:ControlPort
Destination
ClientIP_2:CasualPort
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.31

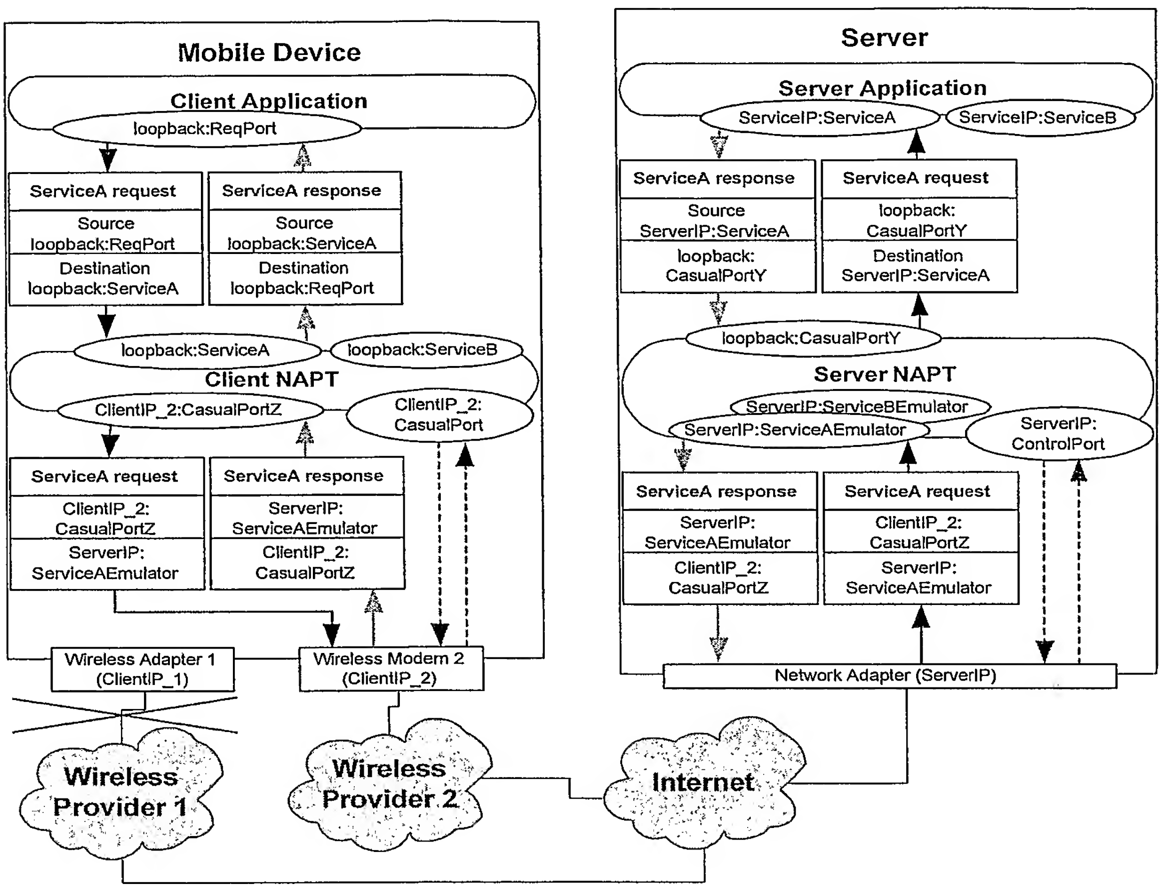

Mobile Device
Client Application
loopback:ReqPort
ServiceA request
Source
loopback:ReqPort
Destination
loopback:ServiceA
ServiceA response
Source
loopback:ServiceA
Destination
loopback:ReqPort
loopback:ServiceA
loopback:ServiceB
Client NAPT
ClientIP_2:CasualPortZ
ClientIP_2:
CasualPort
ServiceA request
ClientIP_2:
CasualPortZ
ServerIP:
ServiceAEmulator
ServiceA response
ServerIP:
ServiceAEmulator
ClientIP_2:
CasualPortZ
Wireless Adapter 1
(ClientIP_1)
Wireless Modem 2
(ClientIP_2)
Server
Server Application
ServiceIP:ServiceA
ServiceIP:ServiceB
ServiceA response
Source
ServerIP:ServiceA
loopback:
CasualPortY
ServiceA request
loopback:
CasualPortY
Destination
ServerIP:ServiceA
loopback:CasualPortY
Server NAPT
ServerIP:ServiceBEmulator
ServerIP:ServiceAEmulator
ServerIP:
ControlPort
ServiceA response
ServerIP:
ServiceAEmulator
ClientIP_2:
CasualPortZ
ServiceA request
ClientIP_2:
CasualPortZ
ServerIP:
ServiceAEmulator
Network Adapter (ServerIP)
Wireless
Provider 1
Wireless
Provider 2
Internet

FIG.32

5